Indexation vidéo par l'analyse de codage

Lionel Brunel

UNIVERSITÉ DE
NICE - SOPHIA ANTIPOLIS

Unité de Formation et de Recherche Sciences
École doctorale : « Sciences et Technologies de l'Information et de la Communication »

THÈSE

pour obtenir le titre de :
Docteur en Sciences
de l'Université de Nice - Sophia Antipolis

Discipline :
Automatique traitement du signal
et des images

présentée et soutenue par :
Lionel BRUNEL

« Indexation vidéo par l'analyse de codage »

Thèse dirigée par Pierre MATHIEU
soutenue le 30 mars 2004

Jury :

Président du jury	Michel BARLAUD	Professeur des Universités Université de Nice - Sophia Antipolis
Rapporteurs	Dominique BARBA	Professeur des Universités Université de Nantes
	Henri NICOLAS	Chargé de Recherche - H.D.R. IRISA/INRIA de Rennes
Examinateurs	Jean-Luc DUGELAY	Professeur à EURECOM EURECOM - Sophia Antipolis
	Pierre MATHIEU	Maître de Conférences Université de Nice - Sophia Antipolis

À mon père bien trop tôt disparu,
À ma mère qui m'a toujours encouragé,
À mon frère Gaëtan,
À ma famille.

« Ah ! se me sabien entèndre !
Ah ! se me voulien segui ! » [1]
Frédéric Mistral

« Ceux qui vous disent que *Linux* et LaTeX ne sont pas conviviaux vous mentent,
je n'ai jamais rencontré autant de gens que depuis que je les utilise... »
Algorithme *Fortune*

[1] « Ah ! si on savait me comprendre ! Ah ! Si on voulait me suivre ! ». Vers gravés sur le socle de la coupe, donnée en 1867 par les Catalans aux Félibres provençaux.

Remerciements

Je tiens à remercier Monsieur le Professeur Gérard Favier, Monsieur le Professeur Pierre Bernhard et Monsieur le Professeur Jean-Marc Fedou, qui ont été les directeurs successifs du laboratoire I3S[2] et qui m'ont permis d'effectuer ma thèse en son sein.

Je tiens à remercier Monsieur le Professeur Michel Barlaud qui m'a accueilli au sein de l'équipe CReATIVe[3] dont il est le responsable. Je tiens à le remercier chaleureusement pour ses aides nombreuses, dont les aides financières, qui m'ont permis de participer à des conférences, mais aussi son aide précieuse lors de la rédaction de mon manuscrit. Grâce à lui, j'ai pu goûter au monde de la recherche.

Je tiens à remercier Monsieur Pierre Mathieu, mon directeur de thèse, sans qui rien n'aurait été possible. Il a su me guider sur le chemin de la recherche. Il a toujours été disponible et m'a permis également de connaître le monde magique des logiciels Libres.

Je tiens à remercier chaleureusement les membres du jury. Tout d'abord Monsieur le Professeur Michel Barlaud qui a accepté d'en être le président ; Monsieur le Professeur Dominique Barba et Monsieur Henri Nicolas qui ont été mes rapporteurs. Je les remercie pour le temps qu'ils m'ont consacré afin de corriger les erreurs, les omissions ou approfondir l'explication de certaines parties. Je tiens à remercier Monsieur le Professeur Jean-Luc Dugelay d'avoir accepté d'être dans le jury.

Je tiens à remercier les autres permanents de l'équipe : Marc Antonini, qui a été mon encadreur de stage de DEA, Eric Debreuve qui m'a aidé lors de la rédaction de certaines parties de mon manuscrit.

Je tiens à remercier le personnel de l'I3S, Madame Micheline Hagnere, toujours dévouée et toujours sympathique.

Je tiens à remercier les deux '*Roots*' de l'I3S : Monsieur Patrick Balestra, toujours prompt à répondre à toutes mes interrogations métaphysiques et Monsieur Lionel Baillergeau qui a été mon mentor dans les réseaux.

Les remerciements sur l'I3S ne seraient être complets sans remercier Monsieur Guy Tessier toujours là afin de rendre service et au caractère très sympathique !

[2] acronyme de Informatique, Signaux, Systèmes de Sophia-Antipolis. C'est un laboratoire qui est une Unité Mixte de Recherche (UMR) entre l'Université de Nice - Sophia Antipolis et le CNRS, sous le numéro administratif 6070.

[3] acronyme de *Compression, REconstruction Adaptés au Traitement d'Images et à la VidEo*, équipe, qui fait partie du groupe *Images Numériques et Vidéo* du laboratoire I3S.

Mes remerciement aussi à tout le personnel de l'I3S que j'ai côtoyé au cours de mes années passées au sein de ce laboratoire.

Si je suis arrivé jusque là, c'est grâce au corps enseignant, et je voudrai en nommer quelques membres : Monsieur Peyron (Professeur de Mathématiques au lycée), Monsieur Hervier (MCF d'informatique à l'UNSA), Monsieur Le-Barz (MCF de mathématiques à l'UNSA) et Feu Monsieur Cerezo (MCF de mathématiques à l'UNSA). Ils ont toujours été pour moi des exemples à suivre.

Au cours de mes années de thèse, j'ai côtoyé de nombreux collègues qui sont devenus des amis au cours du temps.

En tout premier lieu je tiens à remercier Christophe Parisot, car il a, et restera toujours pour moi un موسوعة العلوم, ou une ходячая энциклопедия. Pour ne pas froisser sa modestie, je ne donnerai pas la traduction.

Je tiens à remercier ceux avec lesquels j'ai eu le bonheur (que j'espère partagé !) d'occuper mon bureau. Tout d'abord Stéphane Tramini, qui m'a beaucoup aidé, en particulier, pour l'écriture des articles en anglais. Malgré son tempérament légèrement bougon, il a toujours été présent. Il y a eu Pascal Bouvier, un adorateur de Carcès et un fana de jeux sur console. Cyril Bergeron, mon '*Home Boy*' car pour lui j'étais son '*Root Boy*' (NDR j'étais le *Root* de son ordinateur sous *Linux*). Marco Cagnazzo, mon autre موسوعة العلوم en LaTeX et en C. Avec lui, j'ai passé de grands moments de discussions toujours très intéressants. Les remerciements de mes cobureaux ne seraient pas complets sans citer Eugenia Tardio-Cruz, Tarik Makram et Yanggang Shi.

Je remercie tous les 'non-permanents' de l'équipe CReATIVe. Mes deux seules "*Home Girls*", Stéphanie Jehan qui est arrivée en même temps que moi en DEA ; elle a toujours été une oreille attentive et d'un très bon conseil, ainsi qu'un exemple à suivre ; Annabelle Gouze pour sa présence bienveillante.

Je tiens à remercier Frédéric Précioso ; lui aussi peut recevoir, en ce qui concerne l'informatique, le label ходячая энциклопедия. Également, un grand merci pour ce qui est de l'explication sur les *splines* et la permission d'utiliser son programme. Lui aussi a dû subir la visite de Carcès...

Je tiens à remercier Valéry Valentin qui, avec Christophe Parisot, a eu à subir de nombreuses fois mes essais vocaux. Un grand merci.

Je tiens à remercier Muriel Gastaud, à qui je dois de connaitre quels sont les comportements machos ; Manuela Pereira, qui a su me faire aimer le Portugal ; Ariane Herbulot, qui m'a aidé dans la compréhension de certains problèmes, Frédéric Payan, compagnon d'infortune dans les trains surchauffés italiens ; Tristan Roy, avec qui nous avons souvent parlé politique ; Olivier Amadieu, compagnon de belote ; Thomas André, toujours fervent suporter de l'OM.

Les remerciements sur l'I3S ne seraient pas complets sans les personnes des autres équipes. Diane Lingrand et Johan Montagnat toujours amicaux avec moi. Pascal Rapicault, autre partenaire de belote et maintenant canadien ; Anne Désideri-Bracco, également partenaire de belote et qui a souvent été dans l'obligation de me dire ce qu'il fallait dire et ne pas dire ; Karim Ben Chehida, autre adorateur de Carcès, ce qui est un très grand point positif ; Adeline Capouillez et famille, qui me

taquine toujours ; Xavier Augros, partenaire de belote et papa de Victor et Ilona, Olivier Ponsini qui connait maintenant la vérité sur les téléphones...

Passons maintenant aux aides extérieures à l'I3S. Je tiens à remercier vivement pour leur aide précieuse, Jean-Yves Guérin, mon correcteur orthographique et ami, toujours présent au bon moment ainsi que Christophe Ribuot, toujours là lors des '*Install Party*' ; tous deux toujours prêts à me remonter le moral dans mes moments de faiblesse mais aussi lors des bons moments. Je ne saurai jamais assez les remercier pour toute l'aide qu'ils m'ont apportée.

Je tiens à remercier les relecteurs extérieurs, tout d'abord Christophe Prat, rencontré dans un forum de discussions, et dont le visage m'est inconnu ; Jean-Yves Guérin, qui a remis souvent l'ouvrage sur le métier ; Christophe Parisot qui m'a fait de nombreuses remarques et enfin ma mère. Ils ont été pour moi des atouts précieux lors de la rédaction de ce manuscrit.

Je tiens à remercier Karine Sanche, présence féminine du groupe d'amis, elle m'a aidé (grandement) lors de la rédaction de mes articles en anglais. Malgré ses goûts musicaux, elle a su s'intégrer dans le groupe...

Je tiens à remercier Josué Belliot pour les nombreuses aides qu'il m'a apporté dans la programmation de certaines parties de mon programme.

Je tiens à remercier Benoît Pescheux, qui a toujours été un soutien car adorateur de la Recherche. Merci aux amis, Silvia Pei, modèle de docteur, Jean-Pierre Nebout, qui ne manque pas d'assurance, Philippe Bertaina, enfin des forfaits pour vous simplifier la vie, Olivier Garot, le parisien qui va partir dans l'Est... Ils ont toujours été là.

Je tiens à remercier Michel Parachini pour son beau livre traitant de la couleur ce qui m'a bien aidé.

Je tiens à remercier toutes les personnes présentes et qui ont fait de nombreux kilomètres pour assister à ma soutenance. Merci beaucoup.

Pour terminer je tiens à remercier les moyens techniques de la B.V.F.

Table des matières

Table des figures

Liste des tableaux

Introduction

« L'œil existe à l'état sauvage »
Le surréalisme et la peinture, André Breton (1896-1966)

Résumé : Avec l'apparition de l'ordinateur et l'augmentation de sa puissance, l'homme a pensé qu'il pourrait rapidement mimer la nature, voir même la dépasser. En effet, l'ordinateur est beaucoup plus rapide que l'homme dans tout ce qui est déterministe et certains philosophes pensent que tout dans la nature obéit à des lois rigoureuses, y compris les conduites humaines (« les mêmes causes produisent les mêmes effets ») d'où la possibilité de prévoir, de reproduire un phénomène et enfin le mettre sous forme d'équations que l'ordinateur pourra résoudre très rapidement. Mais ceci n'est vrai que si le système n'a pas une dépendance sensitive des conditions initiales, c'est-à-dire que de petites variations dans les conditions initiales ne produisent que de petites variations dans l'état final du système. D'où le problème ouvert sur la segmentation d'objets en mouvement. Qu'il est donc vexant de ne pas arriver à modéliser ce qu'arrive à faire une mouche !

Désir de mimer la reflexion humaine

Notre cerveau est habitué à traiter en un temps très bref la reconnaissance d'une personne, la distance à laquelle peut être l'objet ou même reconnaître deux objets ayant une couleur similaire. Voici quelques problèmes, pourtant courants, qui impliquent de multiples efforts de recherche dans les sciences de l'informatique, avec pour l'instant des solutions encore incomplètes.

En effet, malgré la constante augmentation de la puissance des calculateurs, malgré les approches théoriques de plus en plus sophistiquées, un certain nombre de tâches résistent encore aux algorithmes et aux méthodes classiques de traitement des signaux et des données. Ces tâches relèvent typiquement du traitement, en temps réel, de très grands flux de données souvent multidimensionnelles et arrivant à des cadences élevées. Le grand nombre des données, leur variabilité, le fait qu'elles ne correspondent pas à des modèles physiques connus nous laissent souvent démunis devant des tâches de caractérisation, de reconnaissance et de prise de décisions. Ce que l'homme veut, c'est pouvoir automatiser ce qu'il sait faire, c'est-à-dire pouvoir résumer une image par une phrase. En effet, l'ordinateur sait donner un sens à une phrase (après un apprentissage). Le nombre de mots étant fini, le nombre de combinaisons l'est aussi, ce qui n'est pas le cas pour les images. Sur les deux images qui suivent[4], nous pouvons dire bien des choses...

FIG. 1 : Cherchons les points communs et les différences

Imiter le cerveau, dans ce domaine, restera pour longtemps encore une ambition démesurée.

Les raisons de l'automatisation

Les techniques numériques appliquées au multimédia sont en pleine expansion. Le volume de données disponibles croît exponentiellement, profitant des progrès technologiques effectués en stockage, en compression et transmission de flux multimédias. Une véritable culture de l'image s'instaure et occupe une grande part dans

[4]la tour de Pise : Jean-Marc Dubré | la tour Montparnasse : http ://www.lemansre.com/

l'espace de communication que l'homme cherche à construire pour mieux communiquer avec son prochain, mieux transmettre de l'information, mieux se divertir, mieux apprendre. Il faut donc que l'ordinateur, afin de faciliter ces échanges, puisse faire de la récupération de données pertinentes ; ainsi il deviendrait un performant outil intelligent de navigation et de recherche dans des systèmes multimédias manipulant de grandes quantités de documents vidéos.

En effet, les moteurs de recherche d'Internet permettent de retrouver dans l'immense bric-à-brac du réseau des réseaux, des informations textuelles. Nous voyons bien que, dans un contexte multimédia, se limiter au texte est trop restrictif. Par exemple, dans nos deux photos précédentes, que pouvons-nous dire textuellement pour les différencier ou les mettre en correspondance ?

Les points communs :

- ce sont des tours (car plus hautes que larges)
- la couleur est ...
- ...

Les différences :

- l'une est penchée tandis que l'autre est droite
- l'une, d'après les motifs de style roman doit être en pierre, l'autre semble plutôt béton-verre
- l'une à l'air vieille et l'autre beaucoup plus récente
- ...

Lors du descriptif, il serait plus facile de dire que la photographie de gauche représente la tour de Pise et celle de droite, la tour Montparnasse. Donc si l'on définit les bonnes clefs pour indexer la photographie de la tour de Pise (bâtiment, style roman, tour, pise...), il est possible de retrouver cette image très rapidement.

Cette évolution permet d'envisager l'automatisation des systèmes de vidéo à la demande mais aussi le traitement en temps réel de vidéo. Par exemple, nous faisons face à une augmentation du nombre de caméras de surveillance, pas seulement sur le domaine privé et il n'est pas possible d'augmenter le nombre de personnes qui regardent et déclenchent des alarmes en fonction des images visualisées (nous pouvons citer les travaux de l'équipe de M. Thonat *et al.* [17] ou l'article de B. Lo *et al.* [63]). Il faut donc arriver à modéliser tous les cas possibles afin qu'en définissant des seuils d'alerte, il puisse être possible de déclencher l'alarme de façon pertinente et c'est en dernier ressort l'opérateur humain qui prendra la décision. Moins le nombre de paramètres est important, plus la compréhension de la séquence est facile pour l'ordinateur. Par exemple prenons la surveillance du trafic routier, la caméra est fixe et de plus nous connaissons le fond de l'image. C'est pour cela qu'il est plus facile d'extraire les objets en mouvement et d'analyser ces derniers (à partir d'un suivi de trajectoire). Grâce à cela, il est possible de donner des alertes (détection d'incidents) comme par exemple la circulation à contresens d'un véhicule ou bien s'il passe sur une portion de route 50 voitures par minute et qu'un kilomètre plus loin, il n'en passe plus que 10 (alors qu'il n'y a pas de sortie possible), on peut déduire un accident entre les deux caméras et immédiatement lancer un processus de vérifications par un opérateur humain. L'intervention humaine sert uniquement à

décider du bien fondé de l'alarme. Il existe déjà des systèmes commerciaux qui font cela, comme Naos, par exemple (équipe B. Neumann au début des années 1990) ou bien Epex en 1988.

Un autre exemple d'utilisation de ces systèmes est la surveillance d'activité humaine. Cela a pour objectif de surveiller des zones à comportements spécifiques, telles que le métropolitain, les parkings, les super-marchés, les aérogares ou bien les banques... Le but est de détecter les comportements anormaux d'individus évoluant dans ces zones et de prévenir les comportements dangereux. Ces comportements correspondent par exemple à des actes de vandalisme ou la préparation de ces actes, mais aussi la vente de drogue ou des agressions. Pour cela, il faut arriver à modéliser le comportement qui sera dit normal et celui dit anormal (devant générer une alarme). Dans ces cas là, la difficulté est immense. Nous pouvons signaler les travaux de M. Coimbra *et al.* [29].

Un autre exemple d'utilisation est l'analyse de scènes sportives. Dans ces applications un système d'interprétation analyse les comportements et mouvements des sportifs. Par rapport au cas précédent, nous avons un environnement plus contraint (*i.e.* un terrain de sport) et un nombre limité de comportements. Les sports concernés sont principalement le football (article de Y. Seo *et al.* [99]) mais aussi le tennis (article de H. Miyamori ou G. Sudhir *et al.* [73, 105]) ; ce dernier sport ayant une structuration du jeu plus marquée. Dans les deux cas, il est donc possible de suivre un joueur, de savoir qui a la balle et avec ces informations, de suivre un match sur un téléphone portable, en simulant le mouvement des joueurs et de la balle.

Cette énumération n'a pas pour but d'être exhaustive, mais de fixer les idées sur les possibilités d'application d'un système d'interprétation et par la suite d'indexation.

Revenons maintenant au traitement de matériel multimédia du style cinéma ou télévision en vue d'une indexation.

L'indexation, $\mathcal{M}$PEG-7 en quelques mots

L'indexation textuelle est normalisée dans la norme $\mathcal{M}$PEG-7 [1] en cours d'élaboration (*cf.* annexe page 146). Cette nouvelle norme, contrairement à ce que pourrait faire croire son nom, ne consiste pas en une septième méthode de compression de données vidéos, mais en un principe très général d'indexation de documents. Il peut certes s'agir de textes, mais c'est surtout au niveau de l'image, la séquence d'images, le son, la musique ou la vidéo que $\mathcal{M}$PEG-7 apportera le plus.

Dans ce mémoire vous trouverez les méthodes mises en place en vue d'indexer le signal vidéo au niveau de séquences vidéos.

Les études préliminaires montrent qu'il est nécessaire d'analyser et segmenter la séquence d'images à la fois dans le temps et dans l'espace. Nous retrouvons les traitements, maintenant classiques, des séquences : prédiction de mouvement ($\mathcal{M}$PEG1-2 [2, 3]), segmentation d'objets ($\mathcal{M}$PEG-4)...

L'axe de recherche que nous proposons est le suivant : à partir de séquences codées au format $\mathcal{M}$PEG1-2 (ou avec d'autres codecs de même type, comme H.261

ou H.264 qui utilisent un partitionnement de l'image en blocs et un suivi temporel de ces derniers), nous utilisons les résultats du codage et surtout de l'analyse de la séquence déjà effectuée pour celui-ci comme point de départ de l'analyse en vue de l'indexation. L'avantage de ce principe d'étude est, dans un premier temps, d'éviter la décompression totale du flux en n'effectuant pas la *DCT* inverse. De plus, nous utilisons l'analyse du mouvement déjà effectuée par le codeur sur la séquence afin d'obtenir rapidement une information pertinente et non supervisée. Utiliser cette information permet la rapidité, mais en contrepartie, nous laisse dépendant de l'estimation faite au niveau du codage (qui peut être plus ou moins pertinente) ; c'est pour cela qu'il faudra mettre en œuvre une estimation de la pertinence des résultats ainsi obtenus afin, le cas échéant, d'utiliser d'autres méthodes, plus longues, mais dont, au final, nous sommes sûrs d'obtenir les résultats recherchés.

Structure du manuscrit

Ce document se divise en trois parties ; deux sont indépendantes, la dernière utilise les deux précédentes.

Dans la première partie, nous désirons estimer soit le mouvement de l'observateur (généralement une caméra), soit celui d'un objet par rapport à lui. Pour cela, nous mettons tout d'abord en équations les sept mouvements possibles de la caméra. Dans un second temps, nous effectuons des simplifications car entre deux images successives le mouvement est généralement faible (faibles rotations, faibles translations et faible zoom) ; de plus, l'ouverture de l'optique est supposée, elle aussi, faible. À partir de ces suppositions, nous pouvons effectuer des simplifications dans l'équation du mouvement. Nous verrons ensuite comment $\mathcal{M}$PEG1-2 traite le mouvement en général.

Dans nos contributions, nous verrons qu'avec le seul mouvement fourni directement dans le flux $\mathcal{M}$PEG1-2, il est possible d'estimer, de façon précise, certains mouvements de la caméra. Pour cela nous définirons le mouvement relatif entre deux images successives. En effet, dans le flux $\mathcal{M}$PEG1-2, nous disposons, pour une image, du mouvement relatif par rapport à une ou deux images servant pour la prédiction (nous les appellerons, dans la suite, des images pivots) ; dans la majorité des cas, cette ou ces images sont plus loin dans le flux. À partir de ce mouvement relatif, il est possible d'estimer le mouvement de la caméra, grâce à une méthode itérative enlevant les vecteurs du mouvement en dehors d'un intervalle de confiance (supposition d'une distribution gaussienne). L'étude des images d'erreur nous permettant d'évaluer la pertinence du résultat.

Dans la deuxième partie, nous désirons segmenter la séquence en zones de couleur uniforme. Partant d'une séquence $\mathcal{M}$PEG1-2, nous verrons quel espace de couleur est utilisé. Comme cette norme utilise la *DCT* et que nous nous sommes fixés de ne pas calculer la *DCT* inverse, nous travaillerons sur une image plus petite (huit

fois plus petite dans les deux axes), évitant ainsi la décompression totale du flux. Le fait de travailler sur une image décimée augmente la vitesse de segmentation, mais en contrepartie, atténue la précision des résultats (nous ne pourrons pas obtenir des résultats au pixel près mais au bloc près).

Dans nos contributions, nous définirons une distance colorimétrique, afin de pouvoir agglomérer des blocs de même couleur ou tout du moins dont l'éloignement colorimétrique sera faible. Dans un premier temps, nous appliquerons cette distance sur l'image en cours d'étude, dans un deuxième temps, nous étendrons notre critère aux images qui lui sont proches temporellement. Afin d'améliorer nos résultats, nous utilisons les travaux, effectués au sein de notre équipe, sur les contours actifs appliqués aux *B-Splines*.

Dans la troisième partie, nous utiliserons les deux parties précédentes, ce qui nous permettra de segmenter les objets en mouvement. Dans un premier temps, nous soutrairons le mouvement de la caméra aux vecteurs mouvements. Dans le cas idéal, les seules zones (de couleur uniforme) qui auront un mouvement non nul seront les zones appartenant aux objets en mouvement. Il sera donc possible de les réunir entre elles, et obtenir ainsi les objets en mouvement. N'étant pas dans le cas idéal, nous utiliserons le traitement itératif que nous avons effectué lors de la première partie pour étiqueter les zones qui ont servi à l'estimation du mouvement comme 'fond' et celles qui ont été rejetées au cours des itérations, car en dehors d'un intervalle de confiance, comme 'objet en mouvement'.

Avec tout cela, il est possible de segmenter les objets en mouvement et de les suivre à travers la séquence.

Nous pouvons alors renseigner certains champs de $\mathcal{M}$PEG7, comme le mouvement de la caméra, la position des objets en mouvement ainsi que leur trajectoire.

Glossaire

« Ce que l'on conçoit bien s'énonce clairement,
Et les mots pour le dire arrivent aisément. »
L'art poétique, Nicolas Boileau (1636 - 1711)

Résumé : De la signification de...

Amplitude : intensité de l'onde.

B : voir Bidirectionnelle.

Bézier (courbes de) : méthode de modélisation vectorielle permettant une manipulation intuitive de lignes et de surfaces arrondies. Cette technique permet de tracer des courbes en utilisant des points comportant des tangentes d'inflexion. La position et le nombre de points, la longueur et l'orientation des tangentes déterminent précisément la forme des courbes. Cette technique a été mise au point par l'ingénieur français Pierre Bézier afin de modéliser les carrosseries automobiles.

Backward (vecteur de mouvement) : c'est un vecteur mouvement qui est utilisé pour la compensation de mouvement à partir d'une image de référence ultérieure dans l'ordre de la diffusion.

Bidirectionnelle : noté *B*, type d'images du *GOP* qui subit une prédiction de mouvement bidirectionnelle afin de réduire la quantité d'informations nécessaires pour la transmission.

Bloc : partie de l'image représentant un carré de huit pixels de côté.

Boom : translation verticale de la caméra ; elle s'élève ou se baisse.

B-Spline : en infographie, système permettant d'adoucir et de contrôler la forme d'une ligne ou d'une surface courbe en manipulant la position d'une multiplicité de points de référence (*cf. Spline*).

BSU : *B-Spline* Uniforme (*cf. Spline*).

CCD : acronyme de *Coupled Charge Device*, soit Dispositif à Transfert de Charges. Il forme le capteur des caméras qui donne un flux de sortie RVB sous la forme d'une tension électrique.

Chrominance : soit Rouge, soit Bleue ; représente la quantité de cette couleur dans cette fréquence.

Compensation de mouvement : utilisation de vecteurs de mouvement pour augmenter l'efficacité de la prédiction des valeurs des pixels. La prédiction utilise des vecteurs de mouvement pour fournir la compensation dans l'image de référence passée ou les images de référence passée et future.

COST 211 : de l'anglais *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services* ; c'est un groupe de recherche européen sur les techniques de réduction des redondances temporelles et l'analyse du contenu pour les services multimédias. Ce groupe utilise plusieurs séquences de travail dont les séquences Stefan Edberg, garde-côtes (*Coast Guard*), Hall ou *Foreman*. Dans notre manuscrit, nous travaillons surtout sur ces séquences.

Couleur : définition d'une sensation visuelle ou, par extension, objet ou lumière pouvant être décrits par des termes comme rouge, vert, blanc, ... La

couleur perçue comme appartenant à une surface dépend de la composition de la lumière qu'elle reflète, du champ visuel environnant et de l'état de l'observateur. La qualité de la couleur comprend le scintillement, le brillant, le chatoiement et d'autres variations de l'apparence d'une surface.

Couleur primaire : l'une des couleurs pouvant être mélangée pour produire une large gamme ; deux d'entre elles ne pouvant jamais en produire une troisième. Dans les mélanges de couleurs additives, le rouge, le vert et le bleu sont primaires. Dans les mélanges soustractifs, le magenta, le cyan et le jaune sont primaires.

DC : acronyme de *Direct Current* soit courant continu ; se trouve dans le coin haut gauche de la *DCT* et représente la valeur moyenne du bloc, soit l'énergie maximale.

DCT : *Discrete Cosine Transformation* ou Transformée en Cosinus Discrète ; technique utilisée dans la compression $\mathcal{J}$PEG ou $\mathcal{M}$PEG1-2... Passe du domaine temporel au domaine fréquentiel.

Dolly : translation de la caméra suivant son axe optique

EDP : Équation aux Dérivées Partielles.

Flot optique : mouvement des régions dans l'image (mouvement apparent).

Forward (vecteur de mouvement) : c'est un vecteur mouvement qui est utilisé pour la compensation de mouvement à partir d'une image de référence précédente dans l'ordre de la diffusion.

Fréquence : nombre de fois par seconde qu'un phénomène périodique se répète. Pour la lumière, la fréquence est de l'ordre de 500 terahertz.

GOP : acronyme de *Group Of Pictures*, soit un groupe d'images, débutant par une image de type *I*, et suivie par une succession de *P* et de *B*.

I : voir Intra.

Image : représentation d'un objet donnée par un système optique. Dans notre manuscrit, nous n'utiliserons que des images en couleur qui peuvent être décrites dans plusieurs espaces de couleur. Par exemple, nous avons le RVB pour un capteur *CCD*, et la Luminance - Chrominance Rouge - Chrominance Bleu pour $\mathcal{M}$PEG1-2.

Intensité : quantité de gris dans une couleur, son degré de clarté par rapport au noir ou son degré d'obscurité par rapport au blanc.

Intra : noté *I* , c'est la première image d'un *GOP* qui est comprimée dans la trame $\mathcal{M}$PEG1-2 comme une image $\mathcal{J}$PEG, elle est décodable directement, car elle ne prend pas appui sur une autre image.

L-Cr-Cb : espace de couleur : Luminance - Chrominance Rouge - Chrominance Bleu.

Longueur d'onde : distance d'une crête du champ électrique à l'autre. La longueur d'onde de la lumière détermine sa teinte.

Lumière : rayon électromagnétique capable de stimuler l'œil pour produire des sensations visuelles. La lumière forme une bande intermédiaire entre le spectre électromagnétique s'étendant entre la lumière violette, d'une longueur d'onde d'environ 400 nm, et la lumière rouge d'une longueur d'onde d'environ 700 nm.

Luminance : quantité de lumière émise à la seconde par une zone depuis chaque unité de surface à angle droit par rapport à la ligne de vision. Dans la décomposition d'une image, cela correspond à une partie monochromatique de l'image, elle s'apparente à une image en niveau de gris.

Luminosité : terme ambigu, signifiant intensité d'une source de lumière ou d'une sensation lumineuse pour la lumière. En physique, s'apparente à la luminance.

Macrobloc : partie de l'image représentant un carré de seize pixels de côté.

Mosaïque (image) : représentation d'une image dont chaque bloc a une luminance constante et chaque macrobloc a les deux chrominances constantes.

$\mathcal{M}$PEG : *Moving Picture Expert Group* , groupe d'experts chargés, dans un premier temps, de mettre au point un format de compression vidéo.

NonIntra : les images qui ne sont pas de type *I* sont des NonIntras. Dans cette catégorie se trouvent les *P* et les *B*.

NTSC : acronyme de *National Television System Commitee*, norme de diffusion télévisuelle. Procédé étatsunien lancé en 1953. Surnommé *Never Twice The Same Color* (jamais deux fois la même couleur), il fait transporter simultanément sur la même onde sous-porteuse les deux signaux de chrominance correspondant à un élément d'image, imposant ainsi à la sous-porteuse une double modulation combinée d'amplitude et de phase. Fonctionne en 525 lignes et 30 images par seconde. Les pays qui l'utilisent : Canada, Japon (NTSC modifié), autres pays d'Asie, Mexique, États-Unis d'Amérique.

P : voir Prédite

PAL : acronyme de *Phase Alterning line*, norme de diffusion télévisuelle. Procédé allemand dû à Walter Bruch (1963 chez Telefunken), améliorant le NTSC : une sous-porteuse à modulation de phase et transmission alternée du $D_b D_r$ du SECAM (voir SECAM). Fonctionne en 625 lignes et 25 images par seconde. Plus économique que le SECAM. À noter, le PAL+ qui permet la diffusion d'images en 16/9 pour les écrans compatibles. Les pays qui l'utilisent : Afrique du Sud, Albanie, Algérie, ex-RFA, Australie, Belgique, Brésil, Danemark, Espagne, Grande-Bretagne, Italie, Luxembourg, Norvège, Pays-Bas, Suède, Suisse, ex-Yougoslavie.

Pan : rotation horizontale de la caméra, soit sur la gauche, soit sur la droite (son trépied ne bouge pas).

Pel : synonyme de *Pixel*.

Pivot (image) : image de référence sur laquelle s'appuie la reconstruction des images de type *P* ou *B*.

Pixel : *Picture Element i.e.* plus petit point (généralement de couleur) qui compose une image, en français on l'appelle Eldim (élément d'image). Le Pixel est utilisé pour définir le pouvoir de résolution des capteurs d'images à *CCD*, des caméscopes, des téléviseurs à critaux liquides. En ce qui concerne les téléviseurs à tube cathodique, depuis que les canons sont disposés en ligne et non plus en triangle, le pouvoir de résolution se définit en nombre de lignes (horizontales et verticales).

PSNR : *Peak Signal to Noise Ratio* soit Pic du Rapport Signal sur Bruit. Le *PSNR* est indépendant de la statistique de l'image.

Prédite : noté *P*, type d'images du *GOP* qui subit une prédiction de mouvement arrière afin de réduire la quantité d'informations nécessaires pour la transmission.

Pureté : voir saturation.

Référence (image de) : image de type *I* ou *P*

Rétine : surface sensible à la lumière sur laquelle les images sont projetées, soit par le cristallin pour l'œil, soit par une lentille optique pour une caméra.

RGB : *Red*, *Green*, *Blue*, *cf.* RVB.

RVB : quantité de Rouge, de Vert et de Bleu de l'image (couleurs primaires). En anglais on trouvera *RGB* .

Saturation : terme inventé par les teinturiers pour décrire la force d'une teinte. En vision, elle est utilisée pour décrire la force ou la vivacité d'une teinte.

SECAM : acronyme de SÉquentiel Couleur À Mémoire, norme de diffusion télévisuelle. Procédé français dû à Henri de France (1911-1986), mis au point par la compagnie française de télévision (filiale de la CSF et de St-Gobain). Présenté officiellement en décembre 1959, breveté, adopté par le *PAL* en 1963 et mis en service en octobre 1967. Cette norme repose sur la transmission simultanée du signal de luminance par l'onde porteuse et d'un seul signal de chrominance par la sous-porteuse, l'autre étant transmis, après, séquentiellement (ces signaux de différence de couleurs sont nommés $D_{_r} = R - Y$ et $D_{_b} = B - Y$, avec Y représentant la luminance, R et B représentant respectivement le Rouge et le Bleu. Le signal transmis est en mémoire et est réutilisé au moment où parvient le signal suivant de façon à disposer, sur chaque ligne, de chacun des deux signaux de chrominance. Le procédé SECAM de balayage alterné n'impose à la sous-porteuse que la modulation de fréquence, cela entraîne une plus grande stabilité de l'image et une meilleure couleur. De plus, cela ne provoque aucun risque d'interférences entre les divers signaux et cela permet son utilisation sans modification des équipements d'émission, de relais et d'enregistrement du

noir et blanc. Le SECAM ne permet pas le son stéréo, il faut pour cela le Nicam. Il fonctionne en 625 lignes. Les pays qui l'utilisent : France, ex-RDA, Arabie Saoudite, Bulgarie, Côte d'Ivoire, Cuba, Égypte, Grèce, Haïti, Hongrie, Iran, Irak, Liban, Luxembourg, Maroc, Monaco, Pologne, ex-Tchécoslovaquie, Tunisie, ex-URSS, Zaïre.

Séquence vidéo : succession d'images arrivant à une cadence prédéfinie (généralement 25 images par seconde en Europe) donnant la sensation du mouvement continu.

SNR : *Signal to Noise Ratio* soit Rapport Signal sur Bruit. C'est le rapport de l'amplitude d'un signal donné et de l'amplitude du bruit, à un instant donné. Le *SNR* s'exprime en décibels (dB).

Spline : mot anglais signifiant "latte" (à cause des lattes utilisées dans l'industrie pour épouser certaines formes). En mathématiques, ce sont des courbes de Bézier composites (courbes polynomiales par morceaux). Il existe des courbes *splines* cubiques, quadriques, cela dépend du degré de dérivabilité.

Teinte : qualité par laquelle une couleur se distingue d'une autre. Toutes les couleurs sont identiques à une ou à un mélange de deux teintes du spectre : rouge, orange, jaune, vert, bleu et violet. Noir, blanc et gris ne possèdent pas de teinte. En physique, la teinte est définie par sa longueur d'onde.

Tilt : rotation verticale de la caméra, soit vers le haut, soit vers le bas (son trépied ne bouge pas).

Track : translation de la caméra sur sa droite ou sur sa gauche.

Vecteur résiduel : différence entre le vecteur pris dans le flux $\mathcal{M}$PEG1-2 et le vecteur idéal (calculé à partir de l'estimation du mouvement de la caméra).

Zoom : consiste à changer la longueur focale pendant que la caméra reste stationnaire.

Première partie
Utilisation des vecteurs déplacement de $\mathcal{M}$PEG1-2 pour l'extraction du mouvement de la caméra

« Je hais le mouvement qui déplace les lignes »
Les Fleurs du Mal (la beauté), Charles Baudelaire 1821-1867

Résumé : Nous étudions dans ce chapitre l'estimation du mouvement de la caméra. Entre deux images, il n'est pas possible d'estimer directement tous les types de mouvements possibles ; dans un premier temps, nous nous restreignons à quatre mouvements. Nous mettons en œuvre la méthode du modèle affine simplifié à quatre paramètres.

Les résultats obtenus, et cela très rapidement, sont assez proches de la réalité (il y a une variation limitée à $\pm 0,5$ pixel) si le mouvement global est inférieur au mouvement maximum utilisé lors du codage. De plus, l'étude de l'image d'erreur nous permet d'évaluer la pertinence du résultat. Si l'image d'erreur est trop éloignée de l'image nulle ou s'il y a trop de macroblocs codés sous forme Intra, nous concluons que le mouvement trouvé n'est pas le bon, et qu'il faut mettre en œuvre d'autres méthodes plus longues mais donnant, à coup sûr, un bon résultat (si nous supposons que les objets en mouvement sont minoritaires dans l'image).

Grâce à tous ces renseignements, il est possible de remplir certains champs de mouvement qu'attend la norme $\mathcal{M}$PEG7 (nous effectuons dans l'annexe page 146 un tour d'horizon de cette norme).

Chapitre 1

Modèle du mouvement apparent de la caméra

Dans ce chapitre nous étudions seulement le mouvement de la caméra. L'objet en mouvement étant supposé plat et minoritaire dans l'image, nous ne prendrons pas en compte son mouvement, par exemple sa rotation sur lui-même (mouvement non planaire FIG. 1.1). En effet, avec les informations qui nous sont données dans le flux $\mathcal{M}$PEG1-2, il ne nous est pas possible d'obtenir ces informations là. Pour cela, il faudrait tout d'abord décompresser la totalité du flux, ce que nous nous sommes interdit de faire, et utiliser ensuite un post-traitement par des méthodes vues dans les articles de M. García et H. Nicolas ou H. Nicolas et C. Labit ou H. Nicolas et F. Denoual [40, 41, 42, 82, 80]. Utilisant le flux $\mathcal{M}$PEG1-2, nous sommes tributaires de la qualité du codeur lorsqu'il y a de l'obscurité ou bien des ombres (article de J. Stauder et H. Nicolas [103]). De même, l'estimation de la distance de l'objet par rapport à la caméra est très difficile à estimer sans calibration préalable (articles de F. Morier *et al.* ou F.X. Martinez *et al.* [75, 69]). De plus, nous sommes dans le cas d'une seule caméra. Dans le cas de deux caméras (vision binoculaire), il est plus facile de trouver le mouvement 2D+t, comme dans l'article de J.L. Dugelay *et al.* ou le livre de O. Faugeras [33, 37].

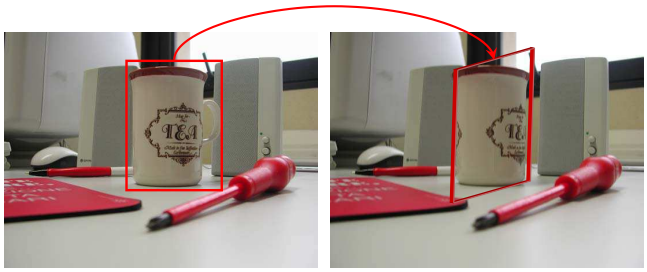
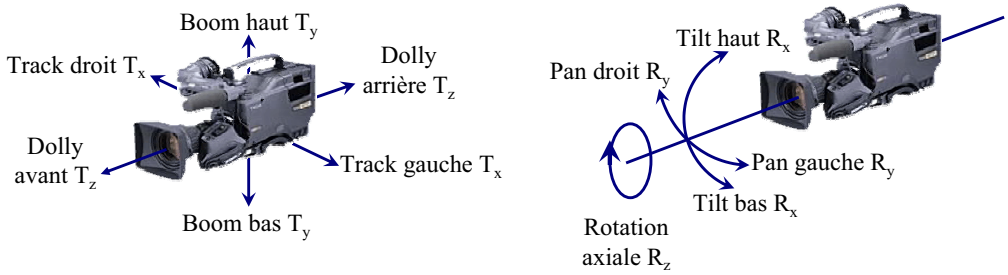


FIG. 1.1 : Rotation non planaire (cas non pris en compte dans notre manuscrit)

1.1 Description du mouvement apparent de la caméra

La caméra peut avoir sept mouvements distincts qui sont :
– trois translations (FIG. 1.2-a) ;
– trois rotations (FIG. 1.2-b) ;
– le zoom.



(a) : Translations possibles (b) : Rotations possibles

FIG. 1.2 : Mouvements possibles d'une caméra

Pour les trois translations, nous avons :

Track : c'est la translation sur la droite ou sur la gauche de la caméra (appelée aussi *travelling* gauche ou droit) ; consiste à déplacer latéralement la caméra.

Dolly : c'est la translation suivant l'axe optique (appelée aussi *travelling* avant ou arrière) ; consiste à reculer ou à avancer (éloigner ou rapprocher) la caméra du sujet. Il donne l'effet de rentrer ou de sortir du personnage.

Boom : c'est la translation verticale, consiste à élever ou à baisser la caméra de haut en bas. Ce mouvement donne un effet de surélévation ou de vol.

À noter que *Track* gauche, *Boom* haut et *Dolly* avant forment un repère orthonormé, dont l'origine est le centre optique de la caméra.

Pour les trois rotations, nous avons :

Pan : rotation horizontale, nommée en français panoramique ; consiste à tourner la caméra de gauche à droite ou de droite à gauche sans bouger le trépied. Alors que si en bougeant la tête rapidement nous pouvons assimiler l'information visuelle, il en n'est rien à l'écran. S'il est trop rapide, ce mouvement ne laisse pas assez de temps au téléspectateur pour assimiler l'information. C'est pour cela qu'il faut être suffisamment lent en effectuant un tel mouvement.

Tilt : rotation verticale ; consiste à pointer la caméra de haut en bas ou de bas en haut sans bouger le trépied. Encore une fois, il est préférable d'effectuer ce mouvement lentement pour que le cerveau humain puisse assimiler l'information.

Axiale : rotation autour de l'axe optique.

Pour le zoom, cela consiste à changer la longueur focale pendant que la caméra reste stationnaire. Ce mouvement n'est pas engendré par un déplacement de la caméra, c'est plutôt une fonction que les caméras possèdent. Faire un zoom avant veut essentiellement dire que nous cadrons le sujet plus serré ; tandis qu'un zoom arrière signifie, par conséquent, que nous cadrons le sujet moins serré.

Dans tout ce qui suit, pour éviter d'utiliser une notation peu usitée, nous conservons la notation anglaise pour les mouvements sauf pour le zoom et la rotation axiale.

Notre but est de trouver l'équation qui caractérise au mieux le mouvement de la caméra. Il faut faire au préalable deux suppositions. Tout d'abord, nous supposons que le mouvement que nous voulons estimer, entre deux images consécutives, est très faible, ce qui veut dire que la translation et l'angle de rotation sont très petits. De plus, en ce qui concerne le zoom, nous supposons, comme H. Nicolas et C. Labit [81], que la distance entre l'ancienne et la nouvelle focale est très petite comparée à la distance par rapport à l'objet. Avec tout cela, nous voyons deux approches du mouvement, qui donnent, au final, une même modélisation.

1.1.1 Équations du mouvement par des développements limités

Modélisation

Pour modéliser l'équation du mouvement, nous considérons que la caméra n'est affectée que d'un seul mouvement sur les sept. De plus, au lieu de faire bouger la caméra, nous bougeons l'objet, ce qui revient au même (dans le repère de l'objet, c'est la caméra qui bouge). Nous considérons deux repères orthonormés :

- $\left(O, \overrightarrow{X}, \overrightarrow{Y}, \overrightarrow{Z}\right)$ avec O le centre optique de la caméra et (OZ) son axe optique ;
- $(o, \overrightarrow{x}, \overrightarrow{y})$ avec o le centre de la rétine ;
- $f = dist(O, o)$ qui est la focale.

Pour simplifier les choses, nous dessinons la rétine de notre caméra dans les $\overrightarrow{Z}$ positifs et non, comme dans la réalité, dans les $\overrightarrow{Z}$ négatifs. Cela permet de simplifier le dessin et d'avoir une image non renversée (FIG. 1.3 page suivante).

Considérons un point $P = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$ dans l'espace ; il se projette sur la rétine R en $p = \begin{pmatrix} x \\ y \end{pmatrix}$. Nous bougeons maintenant P en P' en lui appliquant un seul des six mouvements précités (sauf le zoom). Pour le zoom, nous varions uniquement la focale f. Dans ces sept cas, quel sera le mouvement du point p dans le plan image R ?

Estimons le vecteur de translation $\overrightarrow{u} = \begin{pmatrix} x' - x \\ y' - y \end{pmatrix}$ qui fait passer de p à p'. Le détail des calculs se trouve en Annexe C page 161.

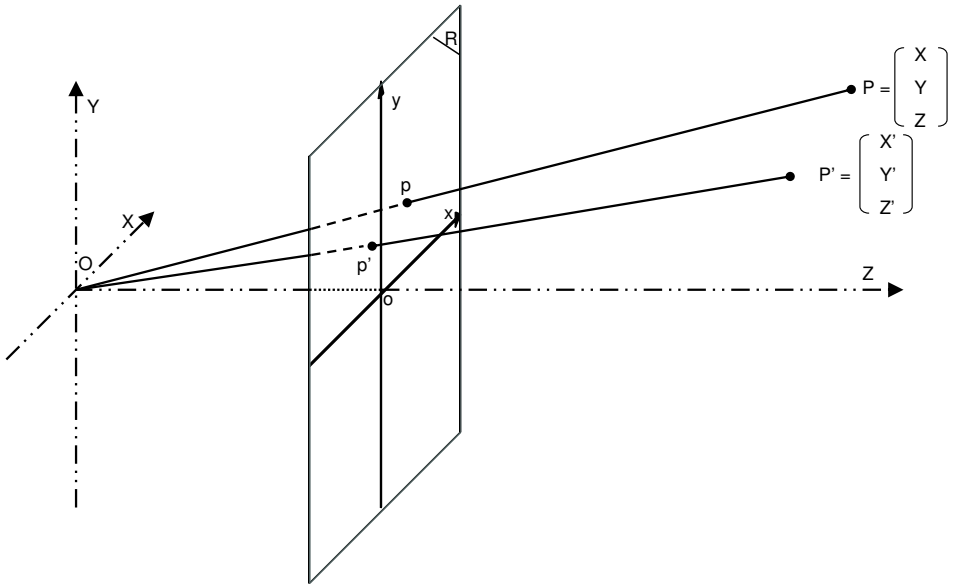


FIG. 1.3 : Repère orthonormé de notre caméra

Nous estimons, dans un premier temps, la nouvelle position de l'objet dans le repère de la caméra (*i.e.* P'), ensuite nous estimons le vecteur directeur $\overrightarrow{u}$ qui nous fait passer de p à p' :

- *Track* (T_x) : $P' = \begin{pmatrix} X + T_x \\ Y \\ Z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} \frac{f}{Z} T_x \\ 0 \end{pmatrix}$
- *Boom* (T_y) : $P' = \begin{pmatrix} X \\ Y + T_y \\ Z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} 0 \\ \frac{f}{Z} T_y \end{pmatrix}$
- *Dolly* (T_z) : $P' = \begin{pmatrix} X \\ Y \\ Z + T_z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} -\frac{x}{Z} T_z \\ -\frac{y}{Z} T_z \end{pmatrix}$
- *Tilt* (R_x) : $P' = \begin{pmatrix} X \\ R_x.Z + Y \\ R_x.Y + Z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} -\frac{x.y}{f} R_x \\ -f.(1 + \frac{y^2}{f^2}).R_x \end{pmatrix}$
- *Pan* (R_y) : $P' = \begin{pmatrix} R_y.Z + X \\ Y \\ R_y.Y + Z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} f.(1 + \frac{x^2}{f^2}).R_y \\ \frac{x.y}{f} \end{pmatrix}$
- Rotation axiale (R_z) : $P' = \begin{pmatrix} R_z.Y + X \\ R_z.X + Y \\ Z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} -y.R_z \\ x.R_z \end{pmatrix}$
- Zoom (R_{zoom}) : $P' = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$, alors $\overrightarrow{u} = \begin{pmatrix} -f.\tan^{-1}\left(\frac{x}{f}\right).\left(1 + \frac{x^2}{f^2}\right).R_{zoom} \\ -f.\tan^{-1}\left(\frac{y}{f}\right).\left(1 + \frac{y^2}{f^2}\right).R_{zoom} \end{pmatrix}$

les rotations sont exprimées en radians et R_{zoom} vaut zéro s'il n'y a pas de modification du zoom.

Équation du mouvement

En joignant tous ces mouvements et en remplaçant le mouvement du point par celui de la caméra, nous obtenons, pour un mouvement complexe :

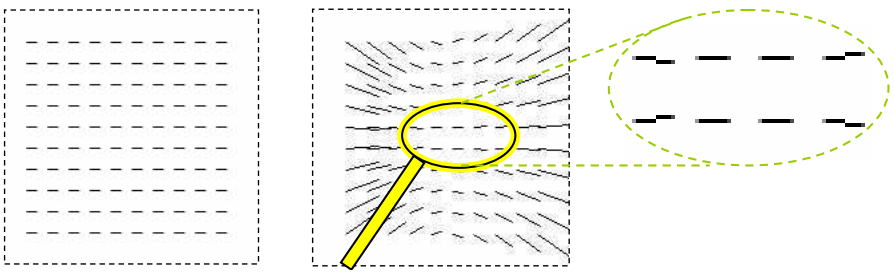
$$\begin{cases} U_x(x,y) = -\frac{f}{Z}.\left(T_x - \frac{x}{f}.T_z\right) + \frac{x.y}{f}.R_x - f.\left(1+\frac{x^2}{f^2}\right).R_y + y.R_z \\ \qquad\qquad + f.\tan^{-1}\left(\frac{x}{f}\right).\left(1+\frac{x^2}{f^2}\right).R_{zoom} \\ U_y(x,y)) = -\frac{f}{Z}.\left(T_y - \frac{y}{f}.T_z\right) - \frac{x.y}{f}R_y + f.\left(1+\frac{y^2}{f^2}\right).R_x - x.R_z \\ \qquad\qquad + f.\tan^{-1}\left(\frac{y}{f}\right).\left(1+\frac{y^2}{f^2}\right).R_{zoom} \end{cases} \tag{1.1}$$

Avec cette équation, pour n'importe quel point de l'image, nous pouvons modéliser de façon théorique les vecteurs de mouvement.

Simplification de l'équation du mouvement à sept inconnues

L'équation sur le mouvement n'étant pas linéaire, il est très difficile de résoudre le système précédent. Pour cela, nous effectuons quelques simplifications qui vont le rendre linéaire, sans pour autant atténuer la pertinence du résultat.

Nous supposons qu'entre deux images consécutives, il n'est pas possible de faire la différence entre le *Pan* et le *Track*. En effet, si nous supposons que nous sommes près de l'axe optique de la caméra, x et y est négligeable, d'où $\frac{xy}{f}$, l'ordonée du *Pan*, est négligeable ; il en est de même pour $\frac{x^2}{f}$. Toujours proche de l'axe optique, nous pouvons approcher l'angle R_y (angle petit) par $\frac{T_x}{Z}$ (en utilisant la tangente). D'où, nous obtenons une équivalence valide pour des déplacements de points proches de l'axe optique (FIG. 1.4) ; ce qui n'est pas le cas sur les bords de l'image.



(*a*) *Track* (*b*) *Pan* avec un grossissement près du centre optique

FIG. 1.4 : Comparaison entre le *Track* et le *Pan*

C'est pour cela qu'il faut supposer que nous sommes dans le cas de petites translations ou de petites rotations entre deux images avec une optique de faible ouverture. En acceptant ces *a priori*, cette approximation est visuellement correcte.

Nous effectuons la même chose pour le *Tilt* et le *Boom*. Grâce à cela, nous pouvons supprimer les termes quadratiques concernant le *Pan* et le *Tilt*.

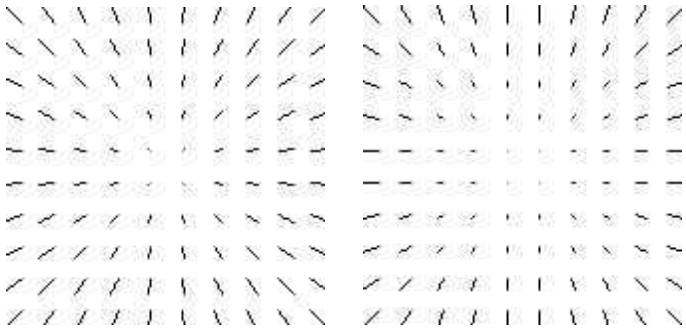
Nous obtenons déjà une équation simplifiée :

$$\begin{cases} U_x(x,y) = -\frac{f}{Z}.\left(T_x - \frac{x}{f}.T_z\right) + y.R_z + f.\tan^{-1}\left(\frac{x}{f}\right).\left(1+\frac{x^2}{f^2}\right).R_{zoom} \\ U_y(x,y) = -\frac{f}{Z}.\left(T_y - \frac{y}{f}.T_z\right) - x.R_z + f.\tan^{-1}\left(\frac{y}{f}\right).\left(1+\frac{y^2}{f^2}\right).R_{zoom} \end{cases} \tag{1.2}$$

Afin de pourvoir approcher T_z à un zoom, il faut le rendre sans dimension (car le zoom représente un grossissement ce qui implique qu'il est sans dimension). Pour cela, nous posons le nouveau *Dolly* qui sera sans dimension (dans la suite de notre document, lorsque nous parlerons de *Dolly*, ce sera ce dernier qui sera utilisé) :

$$\text{nouveau } \textit{Dolly} : \tau_z = \frac{T_z}{f}$$

Avec ce nouveau *Dolly*, nous supposons qu'il n'est pas possible entre deux images consécutives de différencier un Zoom d'un *Dolly* (FIG. 1.5).



(*a*) *Dolly* (*b*) Zoom

FIG. 1.5 : Comparaison entre le *Dolly* et le Zoom

De la même manière que précédemment, il faut supposer que l'ouverture est faible et que la translation suivant l'axe optique ou le zoom est aussi très faible. Pour x et y très proches de zéro, il est possible d'approcher R_{zoom} par τ_z.

De plus, nous supposons être dans le cas d'une caméra à sténopé (sans focale) et dans un modèle orthographique (l'objet est considéré comme plat, ce qui implique une distance constante). Avec ces deux approximations, il est possible d'assimiler $\frac{f}{Z}$ à une constante.

Avec ces deux dernières approximations, nous obtenons l'équation (articles de H. Nicolas et C. Labit ou J. Motsch et H. Nicolas [81, 76]) :

$$\begin{cases} U_x(x,y) = -cte.\left(T_x - x.\tau_z\right) + y.R_z \\ U_y(x,y) = -cte.\left(T_y - y.\tau_z\right) - x.R_z \end{cases} \tag{1.3}$$

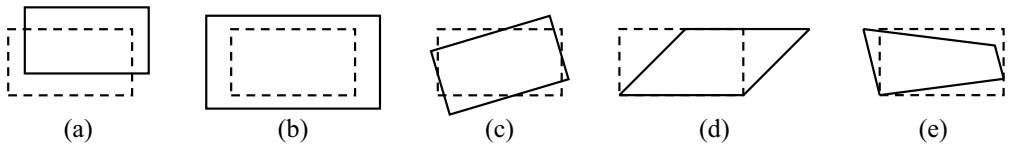
Le résultat ainsi obtenu est linéaire, ce qui va faciliter sa résolution.

Par abus de langage, nous allons dans la suite appeler T_x le *Pan*, T_y le *Tilt*, τ_z le Zoom et R_z la rotation axiale.

1.1.2 Autre approche du mouvement de la caméra (souvent utilisée)

Nous partons d'un champ de vecteurs vitesses, supposés idéaux (représentant tous le même mouvement). Nous verrons un peu plus tard que nous pouvons approcher les vecteurs de mouvement du flux $\mathcal{M}$PEG1-2 comme un champ de vecteurs.

Nous étudions comment retrouver le mouvement de la caméra. Pour cela nous faisons de moins en moins d'approximations, ce qui implique une augmentation du degré de liberté avec un accroissement du temps nécessaire pour les estimations. De plus, nous remarquons que dans ce modèle hiérarchique, les coefficients d'un modèle simple se retrouvent à l'identique dans le modèle immédiatement supérieur (vu dans les articles de C. Stiller *et al.*, P. Migliorati et S. Tubaro ou K. Jinzenji *et al.* [104, 71, 57]). Du fait de la précision des données que nous utilisons, il ne nous est pas apparu pertinent d'essayer d'obtenir des mouvements qui dans les séquences vidéos sont très peu présents. En effet, les valeurs ainsi obtenues, proches de zéro, pourraient être en fait du bruit. C'est pour cela que nous n'aborderons pas, dans ce mémoire, le modèle utilisant la géométrie projective qui permet de donner en plus de la translation, du zoom, de la rotation et du cisaillement, la perspective.



(a) translation - (b) zoom - (c) rotation - (d) cisaillement - (e) perspective

FIG. 1.6 : Les différentes transformations planes.

Dans la figure 1.6, de (a) à (d), ce sont des transformations affines ; pour obtenir la transformation (e), il faut passer dans la géométrie projective. Pour cela, nous renvoyons le lecteur qui le désire à la lecture des livres de O. Faugeras, R. Hartley et A. Zisserman ou B. Horn [37, 46, 48] et à l'article de G. Csurka et P. Bouthemy [31].

Modèle pour un mouvement nul (0 paramètre)

Un champ de vecteurs suit ce modèle lorsqu'il n'y a pas de mouvement. C'est la première approximation. Il est bien sûr facile à estimer et dans un temps nul. Par contre, il est évident que le résultat a peu de chances d'être juste si le mouvement n'est pas nul !

Modèle pour un mouvement apparent de translation (2 paramètres)

Nous supposons ici, que la caméra n'a qu'un mouvement de translation dans le plan perpendiculaire à l'axe optique (soit le *Track* et le *Boom*).

Pour l'estimation :

$$\overrightarrow{u} = \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}$$

avec $\overline{x}$ et $\overline{y}$ les moyennes de tous les vecteurs de mouvement donnés dans le champ.

L'estimation de la moyenne s'effectue très rapidement, mais cela ignore tous les mouvements de rotations, de *Dolly* et de zoom.

Essayons de trouver un modèle un peu plus précis.

Modèle pour un mouvement apparent affine simplifié (4 paramètres)

Ici, nous estimons les mouvements de translations (vus au point précédent), mais en plus, nous estimons le zoom (coefficient d'homothétie) et la rotation axiale. Nous obtenons :

$$\overrightarrow{u} = \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} + \begin{pmatrix} R_{zoom} & -R_Z \\ R_Z & R_{zoom} \end{pmatrix} . \begin{pmatrix} x \\ y \end{pmatrix} \tag{1.4}$$

R_Z représente l'angle en radian de la rotation axiale et R_{zoom} représente le paramètre de divergence.

En posant G le rapport d'homothétie ($R_{zoom} = 0$, lorsque qu'il n'y a pas de zoom) :

$$G = 1 + R_{zoom}$$

nous utilisons l'approximation du premier ordre d'une matrice M qui est une composition de rotation et d'homothétie :

$$M = \begin{pmatrix} \cos(R_Z) & -\sin(R_Z) \\ \sin(R_Z) & \cos(R_Z) \end{pmatrix} . \begin{pmatrix} G & 0 \\ 0 & G \end{pmatrix} = \begin{pmatrix} G.\cos(R_Z) & -G.\sin(R_Z) \\ G.\sin(R_Z) & G.\cos(R_Z) \end{pmatrix}$$

Si nous supposons que l'angle de rotation selon l'axe (OZ) est négligeable (ce qui est généralement le cas) et que de plus le coefficient d'homothétie est proche de l'unité, alors :

$$M - I_2 \approx \begin{pmatrix} R_{zoom} & -R_Z \\ R_Z & R_{zoom} \end{pmatrix}$$

Ce modèle est un compromis entre la vitesse de l'estimation et la pertinence du résultat.

Modèle pour un mouvement apparent affine (6 paramètres)

Par rapport au modèle précédent, il y a des informations de cisaillements (visualisation des effets du cisaillement dans la figure 1.7 page suivante) :

$$\overrightarrow{u} = \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} + \begin{pmatrix} R^x_{zoom} & -R^y_Z \\ R^x_Z & R^y_{zoom} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{1.5}$$

avec R^x_{zoom} ou R^y_{zoom} le coefficient du zoom pour l'axe x ou y respectivement et R^x_Z ou R^y_Z la rotation axiale en radians pour l'axe x ou y respectivement.

Modèle pour un mouvement apparent complexe (modèle quadratique à 12 paramètres)

C'est le même modèle que précédemment, mais nous gardons les termes de second degré. Les termes quadratiques étant difficiles à estimer, ce modèle est très peu utilisé.

À partir le l'équation (1.4) du modèle affine simplifié ou de l'équation (1.3) vue précédemment, si nous connaissons le mouvement de la caméra, il est possible d'estimer pour chaque pixel de l'image à l'instant t, sa position à l'instant $t+1$. Pour notre part, nous effectuons le cheminement inverse. Connaissant le mouvement du pixel de l'instant t à l'instant $t+1$, nous allons estimer le mouvement de la caméra. Le mouvement des pixels nous est donné dans la trame $\mathcal{M}$PEG1-2, étudions maintenant comment ce mouvement est estimé et avec quelles contraintes.

1.2 Traitement du mouvement par $\mathcal{M}$PEG1-2

Ce traitement est généralement connu, mais il nous est apparu nécessaire d'effectuer ce rappel dans le corps du manuscrit et non en annexe, car toute notre méthodologie prend appui sur cette connaissance et nous y ferons sans cesse référence (le lecteur trouvera dans l'annexe A page 141 d'autres compléments sur cette norme).

Dans un film, il existe une forte corrélation temporelle entre images successives ; c'est ce que se propose d'exploiter $\mathcal{M}$PEG1-2. Le codeur décompose le film en une succession de groupes d'images (*GOP* : *Group Of Pictures*) (FIG. 1.8 page 51), qui débute par une Intra (*I*) suivie d'une succession de Prédites (*P*) et de Bidirectionnelles (*B*) intercalées. Généralement, nous avons un *GOP* qui dure un peu moins d'une demi-seconde, soit 12 images (*IBBPBBPBBPBB*) ; cela permet toutes les demi-secondes, de pouvoir se recaler en cas de pertes lors de la transmission et surtout de pouvoir parcourir le film en accéléré. En effet, il est possible d'atteindre directement deux images par seconde décodables directement ; pour les autres, un calcul de compensation de mouvement est nécessaire.

Regardons maintenant les problèmes qui peuvent subvenir, en cas de perte d'informations durant le transfert du flux. Nous les regardons du moins important, pour

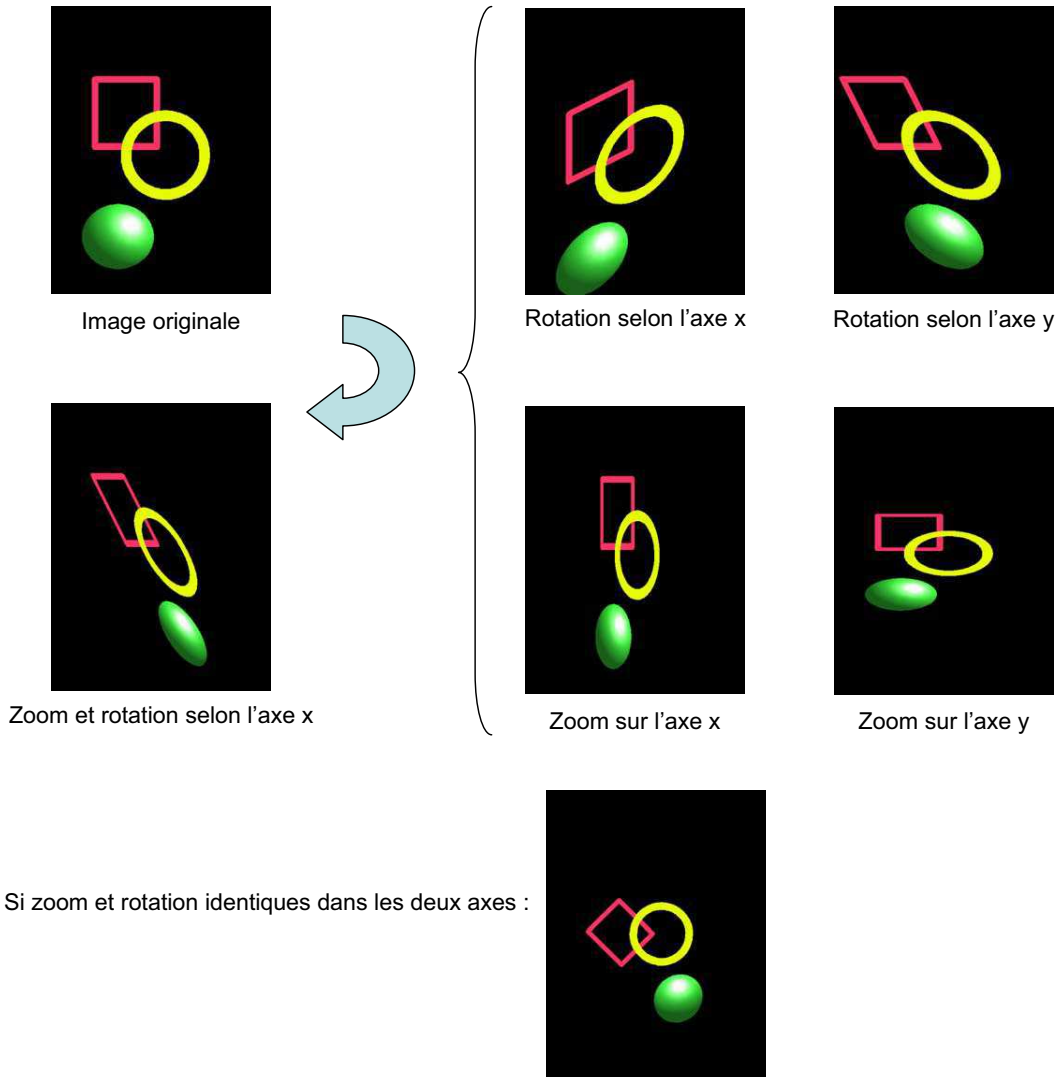


FIG. 1.7 : Si le zoom et la rotation n'affectent qu'un seul axe, ou bien les deux, avec la même valeur

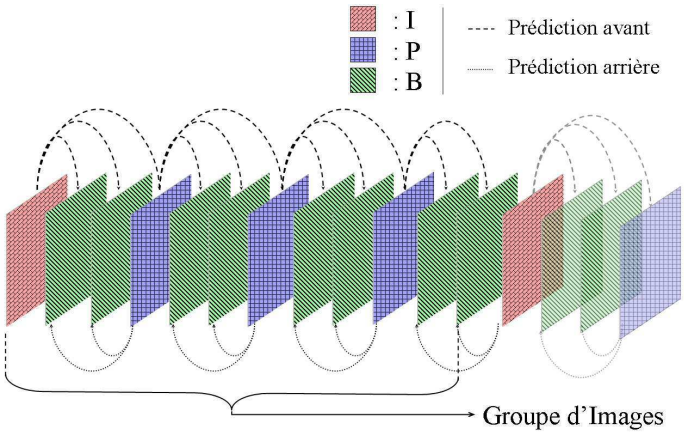


FIG. 1.8 : Composition d'un GOP (MPEG1-2)

la reconstruction de la séquence, au plus important. Sur une image de type *B*, cette perte n'a qu'une incidence localisée au niveau de celle-ci. Sur un *P*, cela entraîne une cascade d'erreurs sur les *B* juste avant et sur toute la fin du *GOP*. Sur un *I*, cela entraîne des erreurs sur la totalité du *GOP*. De plus, si pour le *I* cette perte est totale, cela entraîne l'indécodabilité de tout le *GOP*. C'est pour cela qu'il faut être sûr de ne pas perdre d'informations sur les *I* et le moins possible sur les *P*. Par contre, les *B* n'intervenant dans aucune prédiction, la perte d'informations les concernant serait de moindre importance (un *B* ne représente que $1/25$ème de seconde).

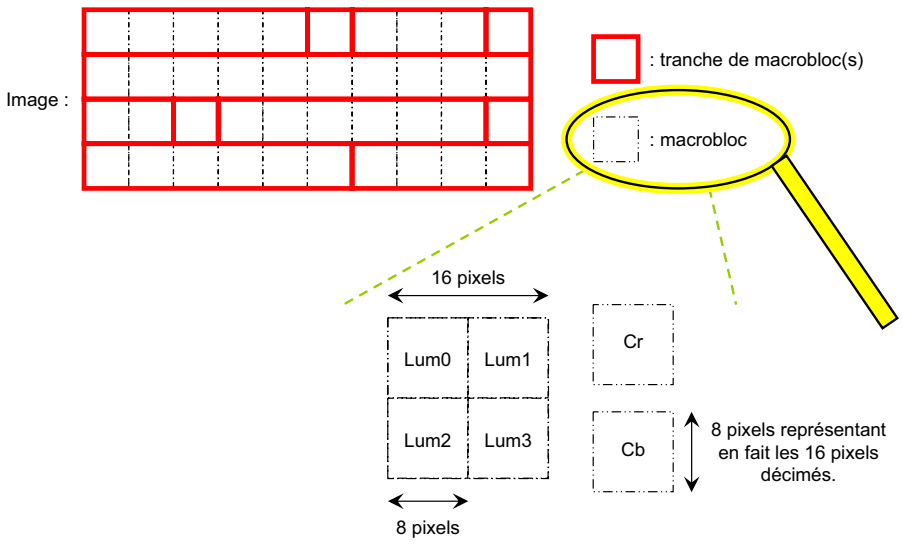


FIG. 1.9 : Découpage d'une image en tranches, macroblocs et blocs

Chaque image (FIG. 1.9) est découpée en tranches de hauteur 16 pixels contenant un ou plusieurs macroblocs. Chaque macrobloc, de dimension 16x16, est composé de six blocs de 8x8 (quatre blocs *L*, un bloc *Cr* et un bloc *Cb*) Nous verrons plus en détails, dans la partie suivante (page 156), la signification des abréviations

L, *Cr* et *Cb* qui représentent la couleur.

Étudions maintenant chaque type d'images plus précisément sous le jour du mouvement :

* Image de type *I* : codage style $\mathcal{J}$PEG (cf. Annexe A page 141), elle est décodable immédiatement, car elle ne prend pas appui sur une autre image comme les autres types d'images.

* Image de type *P* : pour chacun de ses macroblocs, le codeur effectue une prédiction de mouvement à partir du *I* ou du *P* précédant (que nous appellerons image pivot précédente) en y cherchant une zone lui ressemblant (FIG. 1.10 page ci-contre). Cette recherche se déroule dans toutes les directions dans un voisinage spatial, généralement d'amplitude 16 pixels. Si l'on considère qu'entre un *P* et son image pivot (image de type *P* ou *I*) il y a deux images *B* intercalées, cela équivaut à un mouvement maximum entre deux images de moins de 6 pixels. Nous disposons donc d'un vecteur déplacement (appelé vecteur *Forward*).

Cette recherche a pour but de minimiser le poids du macrobloc d'erreur (différence pixel à pixel entre la zone identifiée comme la plus similaire dans l'image *P* ou *I* précédente et le macrobloc que l'on désire coder). La méthode de recherche peut être une méthode exhaustive, ce qui implique un temps de calcul très important, surtout si la zone de recherche est importante. Pour d'autres méthodes nous vous renvoyons à l'article de M. Cagnazzo *et al.* [18]. Si cette mise en correspondance est réussie, l'image d'erreur doit avoir des pixels dont la valeur moyenne est proche de zéro avec un faible écart type. La compression du macrobloc d'erreur est plus forte que pour un macrobloc de type *I*. C'est pour cela qu'il faut vérifier que la reconstruction du macrobloc de *P*, *i.e.* le macrobloc Prédit, avec le vecteur *Forward*, additionné du macrobloc Erreur (qui a subi une forte compression), ne soit pas trop différente de l'image originale. Si tel est le cas, ce macrobloc sera codé sous forme d'un macrobloc Intra. Notons que si le macrobloc d'erreur est très proche du macrobloc nul, son envoi peut être économisé (voir l'exemple page 81).

* Image de type *B* : nous avons deux prédictions, l'une vers l'avant, à partir du *P* ou du *I* précédent (image pivot précédente, comme pour les macroblocs de type *P*) appelée vecteur *Forward* et l'autre, vers l'arrière, à partir du *P* ou du *I* suivant (image pivot suivante), appelée vecteur *Backward* (FIG. 1.10 page ci-contre). Ces deux prédictions utilisent le même algorithme. Le codeur $\mathcal{M}$PEG1-2 estime pour chaque macrobloc, trois images d'erreur :

– celle avec le vecteur *Forward* seul ;
– celle avec le vecteur *Backward* seul ;
– celle avec les deux vecteurs (*Forward* et *Backward*).

Le codeur détermine avec laquelle l'ajout de cette image d'erreur (qui a été compressée) avec sa prédiction va donner l'image la plus proche de l'image originale.

Comme pour les *P*, si à la reconstruction le macrobloc est trop éloigné du macrobloc original, il peut le coder sous forme d'un macrobloc Intra.

Aucun *B* n'étant utilisé comme base de prédiction, cela permet, si le macrobloc d'erreur est 'très proche' du macrobloc nul, de faire l'économie de l'envoi (voir l'exemple page 81).

L'amplitude maximale de recherche vaut généralement 7 pixels dans toutes les directions et cela pour les deux vecteurs (*Forward* et *Backward*). Cela explique que pour le premier *B* de chaque couple, c'est plutôt le vecteur *Forward* qui va être le plus performant et donc celui qui va être le plus envoyé. Pour le second *B* du couple, ce sera le vecteur *Backward*.

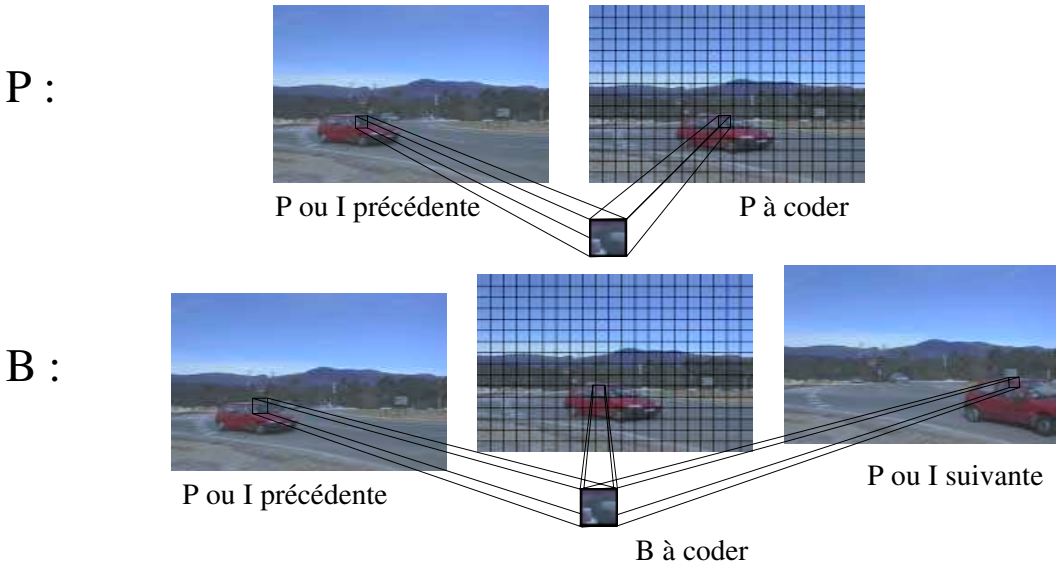


FIG. 1.10 : Prédiction du mouvement sur les *P* et les *B* par rapport à leurs images pivots

1.3 Exemple du traitement du mouvement

Nous allons dans le chapitre suivant utiliser le mouvement et les images d'erreur donnés dans le flux $\mathcal{M}$PEG1-2 en vue de l'estimation du mouvement de la caméra. De ce fait, nous étudions dans les exemples qui vont suivre, tout d'abord comment $\mathcal{M}$PEG1-2 gère les mouvements supérieurs à l'amplitude de recherche. En effet, dans le premier exemple, nous regardons les vecteurs de mouvement fournis dans la trame $\mathcal{M}$PEG1-2 avec leurs amplitudes et leurs directions, qui peuvent varier en fonction de l'amplitude maximale de la prédiction fixée lors du codage. À partir de là, toujours visuellement, nous essayons de déduire le mouvement de la caméra et de l'objet en mouvement. Dans les deux exemples d'après, nous étudions l'impact du débit autorisé sur le poids des images dans des séquences comprimées (poids qui servira à estimer la pertinence de l'estimation du mouvement de la caméra). Ces séquences sont tirées du *COST* 211[1].

Toutes nos séquences $\mathcal{M}$PEG1-2 sont encodées par le graticiel du $\mathcal{M}$PEG *Simulation Group 1994*.

[1] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*

Exemple 1 - Nous codons une séquence routière à caméra fixe. Dans un premier temps nous fixons l'estimation du mouvement à seulement ± 2 pixels entre deux images successives, puis dans l'autre exemple, nous le fixons à ± 6 pixels. La séquence représente une voiture effectuant une translation, de la gauche vers la droite, que nous estimons à environ 5 pixels sur l'image. Regardons de plus près le champ de vecteurs attachés à la voiture.

(a) prédiction jusqu'à ± 2 pixels (b) prédiction jusqu'à ± 6 pixels

FIG. 1.11 : Codage de la même séquence, au même débit, avec une amplitude de prédiction de mouvement différente

Sur l'image de la figure 1.11-b de la présente page, nous remarquons que nous avons bien les vecteurs représentant le mouvement de translation (tous les vecteurs sont parallèles avec une amplitude d'environ 5 pixels). Il y a bien quelques problèmes sur les surfaces vitrées de la voiture de même que sur les zones d'occultations (des zones de routes apparaissent au fur et à mesure que la voiture se déplace), mais ceci est normal, car ces zones n'apparaissent pas à l'identique dans les images antérieures ou postérieures. En ne considérant que cette image de vecteurs, nous pouvons facilement dire que la voiture se déplace dans un mouvement de translation parallèle au plan de l'image et que la caméra est fixe (pas de vecteurs ailleurs que sur la voiture).

Sur l'image de la figure 1.11-a, il n'en est pas de même. En effet, visuellement, que pouvons-nous dire ? Uniquement que la caméra est fixe et que la voiture bouge, quant à son mode de déplacement, rien. Tout au moins, ce n'est pas de cette seule image que nous pouvons déduire quoi que ce soit sur son mouvement.

En conclusion de cet exemple, visuellement, la seule chose commune aux deux flux est que la caméra est fixe. C'est la seule conclusion possible par rapport aux données que nous avons pour le moment. Nous nous rendons compte que nous sommes tributaires de l'amplitude de recherche autorisée par le codeur.

Le mouvement de la caméra sur cette exemple va être bien prédit (car le fond, majoritaire, a un mouvement nul), nous verrons dans la dernière partie (page 117) qu'il est possible d'estimer le mouvement de la voiture grâce au suivi temporel de l'objet en mouvement que nous extrayons.

Exemple 2 - Statistiques sur la séquence Stefan Edberg tirée du *COST* 211 : Cette séquence est de dimensions 352×288 pixels et compte 300 images. Elle va être codée avec des débits dont la consigne varie de 0,54 Mégabits à 12,2 Mégabits par seconde pour des *GOP* de 12 images, au format *IBBPB*...

débit	8,67% de type *I*	25% de type *P*	66,33% de type *B*
$12,2$ Mb/s	9,37% | 1,08	27,24% | 1,09	63,39% | 0,96
$3,0$ Mb/s	16,20% | 1,87	36,78% | 1,47	47,02% | 0,71
$1,0$ Mb/s	21,38% | 2,47	37,14% | 1,49	41,48% | 0,63
$0,54$ Mb/s	18,86% | 2,18	26,73% | 1,07	54,41% | 0,82

TAB. 1.1 : À différents débits de consigne, pour chaque type (avec la séquence Stefan Edberg) : pourcentage du poids total du flux | représentativité $\Re$.

Dans le tableau 1.1 nous affichons, tout d'abord, le pourcentage numérique de chaque type d'images dans la séquence totale. En effet, sur trois cents images il y a 26 images de type *I*, 75 de type *P* et 199 de type *B*. Nous remarquons que le coût de la transmission des informations dans le flux n'est pas le même. En effet, par exemple pour un débit de consigne de 1 Mégabits par seconde, le coût de l'envoi de toutes les images de type *I* est de 21,38% du coût total alors qu'elles ne représentent dans le flux que 8,67% des images.

Cela introduit la notion de représentativité, noté $\Re$, qui est un rapport entre le pourcentage d'images d'un certain type dans la séquence et le pourcentage du poids du flux pour ce même type :

$$\Re = \frac{\frac{P_l}{P}}{\frac{nb_l}{nb}} \quad l \in \{I, P, B\} \tag{1.6}$$

avec :

- P_l : le poids des images de type l dans le flux ;
- P : le poids total du flux ;
- nb_l : le nombre d'images de type l dans la séquence ;
- nb : le nombre total d'images de la séquence.

Par exemple, la représentativité d'un type d'images sera supérieur à un dans le cas où le coût d'envoi de ce type est plus important que sa présence numérique dans le flux.

D'où ici, pour un débit de consigne de 1 Mégabits par seconde, nous arrivons à une sur-représentativité de 2,47 pour les *I* et à une sous-représentativité de 0,63 pour les *B*. Pour le débit de consigne de 12,2 Mégabits par seconde, nous arrivons à une représentativité presque équivalente pour tous les types d'images.

Exemple 3 - Statistiques sur la séquence *Foreman* tirée du *COST* 211, nous faisons exactement la même chose que dans l'exemple précédent.

Ici, nous remarquons, par rapport à l'exemple précédent, que dans tous les cas, les images de type *I* sont en sur-représentation même si le débit de consigne est élevé. En effet, dans cette séquence, le mouvement de caméra est très faible en comparaison du mouvement de la séquence Stefan Edberg. Le codeur arrive sûrement à mieux prédire le mouvement et donc les images d'erreur, dans tous les cas, sont plus faibles.

débit	8,67% de type *I*	25% de type *P*	66,33% de type *B*
8 Mb/s	10,86% ǀ 1,25	27,63% ǀ 1,11	61,51% ǀ 0,93
$3,0$ Mb/s	16,97% ǀ 1,96	37,54% ǀ 1,50	45,49% ǀ 0,69
$1,0$ Mb/s	23,54% ǀ 2,72	39,73% ǀ 1,59	36,72% ǀ 0,55
$0,54$ Mb/s	27,63% ǀ 3,19	37,44% ǀ 1,50	34,93% ǀ 0,53
$0,27$ Mb/s	26,84% ǀ 3,10	24,51% ǀ 0,98	48,65% ǀ 0,73

TAB. 1.2 : À différents débits de consigne, pour chaque type (avec la séquence *Foreman*) : pourcentage du poids total du flux ǀ représentativité $\Re$.

En conclusion de ces deux derniers exemples : Nous constatons que l'envoi d'une image de type *I* est d'autant plus lourd par rapport à l'envoi d'une image de type *B* que le débit de consigne est faible. En effet, c'est sur l'image de type *I* que va se baser toute la prédiction du *GOP* et sur les images de type *P* que va se baser le reste de la prédiction du *GOP*. C'est donc, sur les images *B*, les plus nombreuses en présence mais qui ne servent pas à la prédiction d'autres images, que le codeur effectue la plus grande compression. Comme par la suite nous utiliserons le poids de l'image d'erreur (dans le chapitre suivant pour l'estimation du mouvement de la caméra), il faut garder en mémoire que selon le débit de consigne, le poids peut être plus faible, non à cause d'une meilleure prédiction, mais à cause d'un faible débit de consigne.

1.4 Conclusion

Dans ce chapitre, nous avons vu, dans un premier temps, les éléments qui vont nous être nécessaires pour estimer le mouvement de la caméra. En effet, nous avons repris la modélisation du mouvement de la caméra de deux manières différentes. À partir de simplifications (faibles déplacements entre deux images et petit angle d'ouverture de l'objectif...), il est possible de ne considérer plus que quatre mouvements au lieu des sept initiaux. Ces mouvements sont trois translations et une rotation, soit deux translations dans le plan perpendiculaire à l'axe optique et la troisième parallèlement à cet axe. La rotation est celle autour de l'axe optique.

Dans un deuxième temps, nous avons étudié comment $\mathcal{M}$PEG1-2 utilise la prédiction temporelle dans le but de réduire le débit nécessaire au codage de la séquence. L'étude des exemples nous permet de remarquer que nous sommes tributaires de l'amplitude maximale de recherche et du débit de consigne imposé.

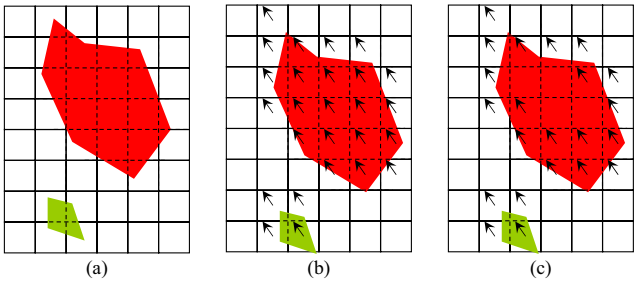
Dans le chapitre suivant, nous utilisons les résultats précédemment obtenus pour estimer le mouvement apparent de la caméra et surtout pour évaluer la pertinence de cette estimation en étudiant, sans la décompresser, l'image d'erreur.

Chapitre 2

Modélisation et estimation du mouvement de la caméra

Plusieurs méthodes ont été proposées pour l'estimation du mouvement de la caméra. La première approche vue dans les articles de N. Aboughazaleh et Y. el Gamal, E. Ardizzone et M. la Cascia ou O. Gerek et Y. Altunbasak [4, 7, 45] est la plus rudimentaire, mais aussi la plus rapide ; elle utilise directement les mouvements qui sont donnés dans le flux de $\mathcal{M}$PEG1-2. Dans [4, 7], le fond est extrait en supposant que tous les macroblocs ayant le même mouvement en font partie. Dans chaque partie, le mouvement de translation est obtenu directement en faisant la moyenne de la totalité des vecteurs de mouvement. Pour obtenir le zoom, l'image est divisée en quatre parties. Le mouvement de translation est estimé, ce qui permet d'avoir une idée du zoom. Un raffinement dans le [45] permet, grâce aux distributions de l'angle des vecteurs, mais aussi de leur amplitude, d'estimer les mouvements de translation et le zoom.

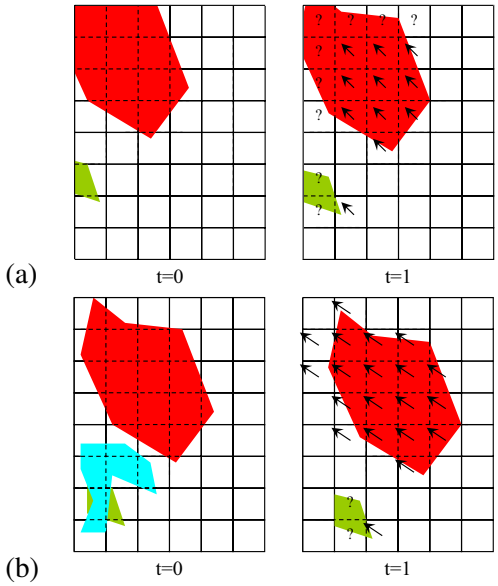
Toutes ces méthodes, qui malgré tout sont très rapides, montrent leurs limites si la valeur des vecteurs de mouvement est bruitée (en direction et en intensité). Cela arrive, par exemple, lorsque il y a de grands aplats de couleur, le codeur ne transmettant alors aucun mouvement (FIG. 2.1) :



(a) : instant initial - (b) : vecteurs idéaux à la suite du déplacement - (c) : vecteurs tels qu'ils peuvent être transmis par $\mathcal{M}$PEG1-2 (le mouvement apparent des zones de couleur uniforme est préféré au mouvement réel).

FIG. 2.1 : Déplacement de deux objets

ou lorsqu'il y a des occultations ou des effets de bord (FIG. 2.2) :



(a) effet de bord : un objet apparaît au fur et à mesure dans le cadre de la caméra ;
(b) occultation : un objet qui était présent et qui cachait une partie des objets en mouvement disparait ou se déplace.

FIG. 2.2 : Effet de bord et occultation :

De plus, ces méthodes n'arrivent pas à gérer les mouvements plus complexes, comme par exemple la rotation selon l'axe optique.

La deuxième approche vue dans les articles de P. Bouthemy *et al.*, J.M. Odobez, C. Stiller et J. Konrad, Y.P. Tan *et al.* ou J. Motsh et H. Nicolas [16, 83, 107, 108, 76], estime une paramétrisation du modèle du mouvement de la caméra. Y.P. Tan *et al.* [108] proposent une méthode rapide pour l'estimation de la translation et du zoom directement dans le flux $\mathcal{M}$PEG1-2. De plus, ils introduisent le rejet des blocs qui sont en dehors d'un intervalle de confiance. P. Bouthemy *et al.* ou C. Stiller et J. Konrad [16, 104] proposent une estimation robuste du modèle affine à six inconnues.

2.1 Modélisation en tout point du vecteur *Forward* normalisé

Les vecteurs *Forward* donnés dans le flux pour le *P* ou les vecteurs *Forward* et *Backward* donnés dans le flux pour le *B* représentent la longueur du déplacement avec l'image, appelée image pivot, qui a servi pour la prédiction (*cf.* page 52). Celle-ci n'est que rarement l'image précédente sauf dans le cas *IPPPP*... et dans le cas du vecteur *Forward* pour le premier *B* de chaque couple dans le cas *IBBPB*... Pour

estimer le mouvement apparent de la caméra entre deux images consécutives, il faut connaître le mouvement d'une image sur l'autre. De plus, pour les images de type Intra, aucun vecteur de mouvement n'est donné (article de R. Wang et T. Huang ou M. V. Srinivasan *et al.* [115, 102]).

Nous avons vu dans le chapitre précédent que l'image est découpée en macroblocs :

- pour les images de type Intra, tous ses macroblocs sont de type Intra, car ils n'utilisent aucune prédiction ;
- pour les images de type *P*, deux possibilités :
  * il y a une mise en correspondance avec l'image pivot précédente ; dans ce cas là, le macrobloc est de type *P* et nous avons un vecteur *Forward* attaché à celui-ci ;
  * il n'y a pas de mise en correspondance ; le macrobloc est codé sous forme d'un macrobloc Intra.
- pour les images de type *B*, quatre possibilités :
  * il y a une mise en correspondance avec uniquement l'image pivot précédente ; dans ce cas là, le macrobloc est de type *P* et nous avons un vecteur *Forward* attaché à celui-ci ;
  * il y a une mise en correspondance avec uniquement l'image pivot suivante ; dans ce cas là, le macrobloc est de type *B* et nous avons un vecteur *Backward* attaché à celui-ci ;
  * il y a une mise en correspondance avec les images pivots précédente et suivante ; dans ce cas là, le macrobloc est de type *Bidirectionnel* et nous avons deux vecteurs, *Forward* et *Backward*, attachés à celui-ci ;
  * il n'y a pas de mise en correspondance ; le macrobloc est codé sous forme d'un macrobloc Intra.

À partir de là, nous supposons que le mouvement de la caméra est continu sur une suite de quelques images ; il est donc possible d'interpoler le ou les vecteurs de mouvement donnés dans le flux pour se ramener à un mouvement par rapport à l'image précédente ; c'est ce que nous nommons vecteur *Forward* normalisé attaché au macrobloc en position spatiale (i, j), noté $\overrightarrow{F_n^{(i,j)}}$.

Nous obtenons donc :

- Cas d'un macrobloc *Intra* pour une image *Intra* : $\overrightarrow{F_n^{(i,j)}} = \frac{\overrightarrow{F_s^{(i,j)}} - \overrightarrow{B_p^{(i,j)}}}{2}$

$\overrightarrow{F_s^{(i,j)}}$: vecteur *Forward* attaché au macrobloc en position spatiale (i, j) de l'image suivante ;

$\overrightarrow{B_p^{(i,j)}}$: vecteur *Backward* attaché au macrobloc en position spatiale (i, j) de l'image précédente.

- Cas d'un macrobloc de type *P* : $\overrightarrow{F_n^{(i,j)}} = \frac{\overrightarrow{F^{(i,j)}}}{nb+1}$

$\overrightarrow{F^{(i,j)}}$: vecteur de mouvement donné dans le flux pour le macrobloc en position spatiale (i, j) ;

nb : nombre d'images entre celle où se trouve ce macrobloc de type *P* et l'image pivot précédente ayant servi à la prédiction.

– Cas d'un macrobloc de type ***B*** : $\overrightarrow{F_n^{(i,j)}} = -\frac{\overrightarrow{B^{(i,j)}}}{nb+1}$

$\overrightarrow{B^{(i,j)}}$: vecteur de mouvement donné dans le flux pour le macrobloc en position spatiale (i, j) ;

nb : nombre d'images entre celle où se trouve ce macrobloc de type *B* et l'image pivot suivante ayant servi à la prédiction.

– Cas d'un macrobloc de type ***Bidirectionnel*** : $\overrightarrow{F_n^{(i,j)}} = \frac{\frac{\overrightarrow{F^{(i,j)}}}{nb_F+1} - \frac{\overrightarrow{B^{(i,j)}}}{nb_B+1}}{2}$

$\left(\overrightarrow{F^{(i,j)}}, \overrightarrow{B^{(i,j)}}\right)$: vecteurs de mouvement donnés dans le flux pour le macrobloc en position spatiale (i, j) ;

nb_F : nombre d'images entre celle où se trouve ce macrobloc de type *Bidirectionnel* et l'image pivot précédente ayant servi à la prédiction ;

nb_B : nombre d'images entre celle où se trouve ce macrobloc de type *Bidirectionnel* et l'image pivot suivante ayant servi à la prédiction.

Lorsque pour un macrobloc appartenant soit à une image de type *P*, soit à une image de type *B*, il n'y a aucune information de mouvement (*i.e.* le macrobloc est envoyé sous forme d'un macrobloc Intra), nous ne mettons en œuvre aucun algorithme pour essayer de déterminer son mouvement. En effet, ce manque d'informations constitue une information en lui-même. Cette absence implique soit que le mouvement a été supérieur au seuil maximal autorisé par le codeur, soit qu'il s'agit d'une partie qui est apparue (après une occultation). Nous verrons par la suite que ces situations adviennent souvent dans les zones où se situe l'objet en mouvement.

2.2 Application aux vecteurs de $\mathcal{M}$PEG1-2

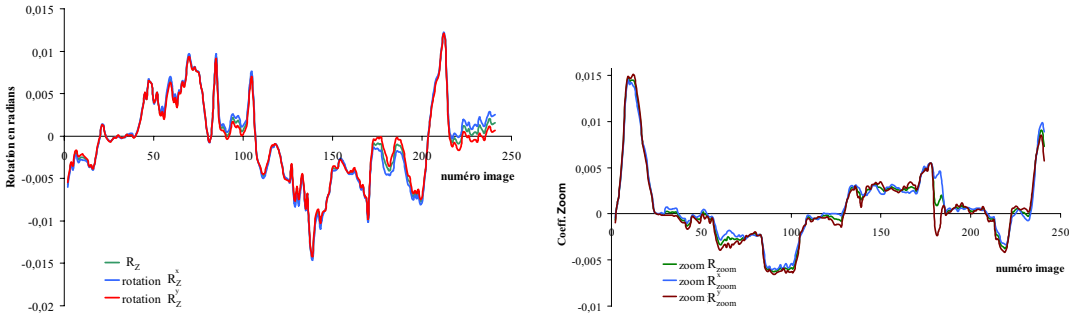
Notre but est d'extraire directement le mouvement du flux donné par le codeur $\mathcal{M}$PEG1-2. L'optique du codeur est différente de la nôtre. En effet, le codeur a estimé le vecteur de mouvement dans le seul but de minimiser l'image d'erreur (FIG. 2.1 et 2.2), de ce fait, il ne représente pas obligatoirement le véritable mouvement. De plus, les vecteurs de mouvement sont modifiés pour améliorer des performances comme le taux de transfert. Enfin, le codeur n'étant pas normalisé, il lui est laissé toute latitude pour l'estimation des vecteurs de mouvement. Il peut, par exemple, les 'estimer' en leur affectant une valeur aléatoire. Cette dernière méthode permet d'accélérer la phase de recherche du bloc similaire dans l'image de référence avec en contrepartie, pour une qualité comparable, un débit nécessaire plus important. Si nous sommes dans ce cas là, l'estimation du mouvement de la caméra, qui s'appuie sur ces vecteurs, sera complètement fausse.

Pour notre part, nous désirons les véritables vecteurs de mouvement qui ont une certaine cohérence avec leurs voisins. Pour cela, il faut évaluer la pertinence des vecteurs résultats du codeur pour, le cas échéant, mettre en œuvre d'autres techniques aux résultats plus exploitables quoique plus longs. Pour cela, il faudrait décompresser complètement le flux et utiliser d'autres méthodes de prédictions de mouvement qui prennent plus en compte le mouvement réel de la caméra comme par exemple dans l'article de J. Wang et D. Barba [114].

2.2.1 Estimation du mouvement de caméra

Tout d'abord, nous effectuons l'estimation avec tous les vecteurs de mouvement donnés par $\mathcal{M}$PEG1-2. Nous nous cantonnons, pour l'instant, au système à quatre paramètres. En effet, le système affine simplifié a un bon rapport qualité - rapidité et lors de nos différentes expérimentations, le passage au modèle affine à six paramètres a engendré une augmentation très nette du coût de l'estimation pour une modification à peine notable du résultat.

Dans la figure suivante nous comparons les résultats obtenus à partir d'un même champ de vecteurs (estimé à partir de la séquence Stefan Edberg) avec deux algorithmes, le premier considérant le modèle affine simplifié, le second le modèle affine non simplifié.



pour la rotation pour le zoom

FIG. 2.3 : Comparaisons entre les modèles affine simplifié et non simplifié pour la séquence Stefan Edberg.

Nous pouvons constater que les variations sont très faibles et la variation maximale représente un décalage maximum sur les bords de l'image (de dimension identique à celle de Stefan Edberg) de un pixel, ce qui est négligeable par rapport à la précision des données (demi-pixel près).

Sur les autres séquences que nous avons testées, nous obtenons toujours, au maximum, le même décalage et le fait de rajouter de nouveaux paramètres à déterminer peut augmenter les instabilités.

L'approche par le modèle affine simplifié (vu page 48) ou l'approche avec l'équation du mouvement à quatre paramètres (vu page 46), aboutissent au même résultat. Pour notre part, nous utilisons l'équation (1.3) page 46. Nous effectuons

une minimisation des moindres carrés entre la partie observée et la partie modèle de mouvement :

$$J = \sum_{i=0}^{N-1} \left\{ \left[U_i^x - \mathcal{F}_i^x \right]^2 + \left[U_i^y - \mathcal{F}_i^y \right]^2 \right\} \tag{2.1}$$

avec :

- l'indice i représentant le i^{eme} macrobloc de la rétine qui en comporte N, il se trouve, dans le repère orthonormé de la rétine, en position (x_i, y_i) ;
- la partie observée $\left(\mathcal{F}_i^x, \mathcal{F}_i^y\right)$ représentant le vecteur *Forward* normalisé $\overrightarrow{F_n^{(x_i,y_i)}}$ pour le i^{eme} macrobloc ;
- la partie dont nous voulons estimer les paramètres $\left(U_i^x, U_i^y\right)$ et qui suit l'équation vu page 46.

Nous avons vu que le vecteur pouvait être transmis ou non, selon le poids de l'image d'erreur. Si pour une image de type *P* ou de type *B*, le macrobloc est codé sous forme d'une Intra alors celui-ci ne rentrera pas dans l'estimation de la minimisation des moindres carrés (dans toutes les sommes sur i variant de 0 à N, nous sauterons les i dont le type du bloc est Intra).

Nous dérivons l'équation (2.1) par rapport aux quatre inconnues qui sont les deux translations T_x, T_y, représentant respectivement dans notre notation le *Pan* et le *Tilt*, τ_z le zoom (*cf.* abus de langage page 47) et la rotation axiale R_z. Au préalable, afin de simplifier les notations, nous posons :

- $S_x = \sum x_i$ et $S_y = \sum y_i$;
- $C_x = \sum x_i^2$ et $C_y = \sum y_i^2$;
- $\mathcal{F}_x = \sum \mathcal{F}_i^x$ et $\mathcal{F}_y = \sum \mathcal{F}_i^y$;
- $\mathcal{L}_x = \sum \left(x_i.\mathcal{F}_i^x\right)$ et $\mathcal{L}_y = \sum \left(y_i.\mathcal{F}_i^y\right)$;
- $M_x = \sum \left(x_i.\mathcal{F}_i^y\right)$ et $M_y = \sum \left(y_i.\mathcal{F}_i^x\right)$;
- n le nombre de macroblocs valides, *i.e.* qui ne sont pas codés sous forme d'Intra, d'où $n \leq N$.

Avec tout cela et après calcul, nous obtenons :

$$\frac{\delta J}{\delta T_x} = 2.\frac{f}{Z}\left[n.\frac{f}{Z}.T_x - \frac{f}{Z}.S_x.\tau_z - S_y.R_z - \mathcal{F}_x\right]$$

$$\frac{\delta J}{\delta T_y} = 2\frac{f}{Z}\left[n.\frac{f}{Z}.T_y - \frac{f}{Z}.S_y.\tau_z + S_x.R_z - \mathcal{F}_y\right]$$

$$\frac{\delta J}{\delta \tau_z} = 2\frac{f^2}{Z^2}\left[-S_x.T_x - S_y.T_y + \left(C_x + C_y\right).\tau_z + \frac{Z}{f}.\left(\mathcal{L}_x + \mathcal{L}_y\right)\right]$$

$$\frac{\delta J}{\delta R_z} = 2.\left[-\frac{f}{Z}.S_y.T_x + \frac{f}{Z}.S_x.T_y + \left(C_x + C_y\right).R_z - \mathcal{L}_x + \mathcal{L}_y\right]$$

Dans la minimisation des moindres carrés, nous supposons que les quatre dérivées partielles sont égales à zéro afin de trouver un *extremum* :

$$\frac{\delta J}{\delta R_z} = \frac{\delta J}{\delta T_x} = \frac{\delta J}{\delta T_y} = \frac{\delta J}{\delta \tau_z} = 0$$

Nous avons donc à résoudre un système. Posons :

$$M = \begin{pmatrix} n\frac{f}{Z} & 0 & -\frac{f}{Z}S_x & -S_y \\ 0 & n\frac{f}{Z} & -\frac{f}{Z}S_y & S_x \\ -S_x & -S_y & C_x + C_y & 0 \\ \frac{f}{Z}S_y & \frac{f}{Z}S_x & 0 & C_x + C_y \end{pmatrix}$$

alors :

$$M. \begin{pmatrix} T_x \\ T_y \\ \tau_z \\ R_x \end{pmatrix} = \begin{pmatrix} \mathcal{F}_y \\ \mathcal{F}_x \\ -\frac{Z}{f}.\left(\mathcal{L}_x + \mathcal{L}_y\right) \\ \mathcal{L}_x - \mathcal{L}_y \end{pmatrix}$$

alors :

$$\begin{pmatrix} T_x \\ T_y \\ \tau_z \\ R_x \end{pmatrix} = M^{-1}. \begin{pmatrix} \mathcal{F}_y \\ \mathcal{F}_x \\ -\frac{Z}{f}.\left(\mathcal{L}_x + \mathcal{L}_y\right) \\ \mathcal{L}_x - \mathcal{L}_y \end{pmatrix} \tag{2.2}$$

2.2.2 Amélioration du résultat

Nous avons fait l'estimation directement avec tous les vecteurs de l'image. Cela suppose qu'il n'y ait pas d'objets en mouvement dans l'image et que les vecteurs existants de tous les macroblocs soient pertinents. La probabilité que cela arrive est très faible, surtout s'il y a des objets en mouvement dans la séquence. Nous avions déjà signalé que le fond devait être majoritaire pour la bonne marche de notre algorithme.

En effet, puisque nous estimons le mouvement majoritaire, si l'aire de l'objet en mouvement devenait plus grande, ce ne serait plus le mouvement de la caméra que nous estimerions, mais au contraire le mouvement de l'objet par rapport à la caméra. Pour résoudre ce problème, il est possible d'utiliser un historique de la séquence en supposant que soit au début du plan, soit à la fin, ce sera le fond qui sera majoritaire.

Nous estimons grâce au système (2.2), le mouvement en première approximation de la caméra. Celui-ci nous permet d'estimer les vecteurs de mouvement idéaux. Nous soustrayons ces derniers aux vecteurs *forward* normalisés, ce qui constitue des vecteurs résiduels. Si notre résolution du système donne le mouvement exact de la caméra et que nous nous trouvons dans le cas idéal, les vecteurs résiduels vont tous être nuls. N'étant pas dans le cas idéal, seule la moyenne de

ces vecteurs résiduels est proche de zéro (si ce n'est pas le cas, c'est que l'objet en mouvement est plus important, ce qui va amoindrir, voire fausser, la précision du résultat). Nous supposons que nous sommes face à une loi Gaussienne. Nous regardons indépendamment l'abscisse et l'ordonnée et nous rejetons les macroblocs ayant l'une des deux en dehors de l'intervalle de confiance que nous fixons à 90%. Pour les autres, nous les marquons comme pertinents.

Nous recommençons l'estimation du mouvement avec le système (2.2), en utilisant les seuls macroblocs ayant été marqués comme pertinents. Nous répétons cette opération et nous arrêtons lorsque plus de 90% des vecteurs restants sont dans l'intervalle de confiance ou bien lorsque le nombre de vecteurs de mouvement restants est inférieur à 30% du nombre initial. Nous sommes sûrs que cet algorithme s'achève, car à chaque fois, nous enlevons des vecteurs de mouvement. Un autre algorithme, qui utilise l'estimation robuste, peut être vu dans l'article de M. Durik et J. Benois-Pineau [34]. Dans cet article, au cours des itérations, chaque vecteur a un poids qui lui est associé et qui varie (il peut augmenter puis diminuer ou l'inverse) alors que dans notre cas, les poids qui au départ valent un peuvent prendre la valeur zéro, et lorsque le poids devient égal à zéro, il n'est plus possible qu'il reprenne la valeur un. Notre méthode nécessite plus d'itérations en faisant très attention à ne pas rejeter des vecteurs mouvements qui sont pertinents ; en effet, ils ne pourront pas re-entrer dans l'estimation. Par contre, notre méthode est plus facile à mettre en œuvre, et constatant que les résultats sont proches de la méthode de référence (vue dans le paragraphe 2.2.3), il ne nous est pas paru utile de programmer la méthode robuste.

Ci-contre, l'algorithme de l'estimation du mouvement de la caméra en pseudo-code.

Algorithme 1 Estimation du mouvement de la caméra pour une image :

```
marquons tous les macroblocs nonIntras comme valides
faire
   pour tous les macroblocs valides faire
      estimer le mouvement de la caméra grâce à
        l'équation (2.2)
   finpour
   calculer pour chaque macrobloc valide son vecteur résiduel
   tracer l'histogramme des vecteurs résiduels
   marquer comme nonValide les macroblocs qui ont un vecteur
     résiduel en dehors de l'intervalle de confiance
tant que (nombre de macroblocs valides >
   30% du nombre total des macroblocs) et
         (nombre de macroblocs valides actuels <
   90 % nombre de macroblocs valides au tour précédent)
rendre le dernier mouvement de la caméra
mémoriser les macroblocs marqués comme non valides
```

2.2.3 Premiers résultats expérimentaux

Nous comparons les résultats obtenus à partir de notre méthode appliquée à un flux $\mathcal{M}$PEG1-2 et une méthode dite de référence.

La méthode de référence, mise en œuvre au sein de notre équipe, estime le champ de vecteurs d'une image par rapport à la précédente en découpant l'image à prédire en blocs de 4×4 pixels afin de minimiser le bloc d'erreur résultant. Cette recherche, dans un voisinage dont nous pouvons fixer les limites, est exhaustive et ce fait au tiers de pixel. À partir de ce champ de vecteurs, une méthode affine simplifiée est utilisée afin d'estimer le mouvement de la caméra. Cette recherche n'utilise pas tous les artifices de programmation qui permettent de trouver dans un temps beaucoup plus bref le bloc à mettre en correspondance. Pour cela, il est possible de trouver plusieurs méthodes d'estimation du flux optique plus rapides dans l'article de J.L. Barron *et al.* [9] qui sont soit basées sur l'étude de la différence, soit sur la corrélation, soit sur l'énergie, enfin sur la phase.

Sur la séquence Stefan Edberg

Nous appliquons notre méthode à la séquence Stefan Edberg (séquence de 300 images de dimensions 352×288 pixels) qui fait partie des séquences de travail données par le *COST* 211[1]. Nous l'avons choisie pour son mouvement de caméra rapide et complexe. De plus, l'objet en mouvement (Stefan Edberg) se déforme et bouge lui aussi grandement. Ce mouvement très important nous permet de tester la fiabilité de notre méthode et, le cas échéant, d'explorer des pistes afin de reconnaître automatiquement que le résultat est faux. Nous utilisons deux flux $\mathcal{M}$PEG1-2 codés avec le graticiel du $\mathcal{M}$PEG *Simulation Group 1994* avec un débit de consigne de 1,1 Mbits/s et une recherche revenant à une prédiction de mouvement limitée à sept pixels entre deux images et à trois pixels pour le deuxième flux.

Nous ne regardons que les images de 14 à 160, en effet, après l'image 160 le mouvement devient encore plus important et donc notre méthode ne fonctionne plus. Vous trouverez dans l'annexe page 169 le *Pan* pour les images de la première à l'image 250. Il est à noter qu'avec notre méthode de référence, nous n'avons pas pu trouver le mouvement sur les cinquante dernières images de la séquence. En effet, sur celles-ci, le mouvement est très rapide et la partie du fond qui est perpendiculaire à l'axe optique de la caméra est très faible et fortement texturée.

Les problèmes d'estimation : entre les images 14 à 160, nous comparons notre méthode rapide avec la méthode dite de référence. Dans cette dernière, il faut au préalable définir la zone de recherche. Plus cette zone est grande, plus l'estimation va être longue. Nous définissons une zone de recherche de 15 pixels entre deux images consécutives et ceci dans toutes les directions. Cette zone est très étendue mais nécessaire pour cette séquence. Une aire de recherche aussi importante augmente de fait le temps de l'estimation. La comparaison du temps nécessaire pour

[1] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*

donner les résultats vus dans la figure suivante et dans l'annexe page 167, donne l'avantage à notre méthode. Pour la séquence sus-mentionnée il faut (sur une machine *Intel* PIII 480 Mhz) plus de dix heures pour la méthode dite de référence et moins d'une vingtaine de secondes pour notre méthode. Il faut rappeler que la programmation des deux méthodes n'a pas été optimisée.

La courbe ci-dessous représente le mouvement du *Pan* , pour les autres mouvements, nous renvoyons le lecteur à l'annexe page 167.

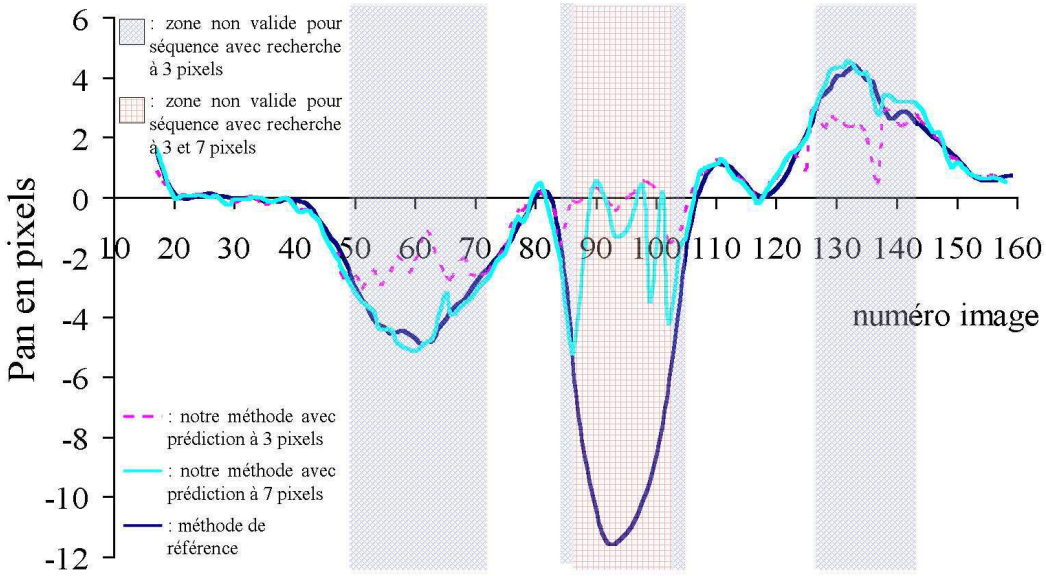


FIG. 2.4 : Comparaisons entre notre méthode (avec deux flux $\mathcal{M}$PEG1-2 : prédiction à 3 pixels et à 7 pixels) et la méthode de référence : estimation de T_x

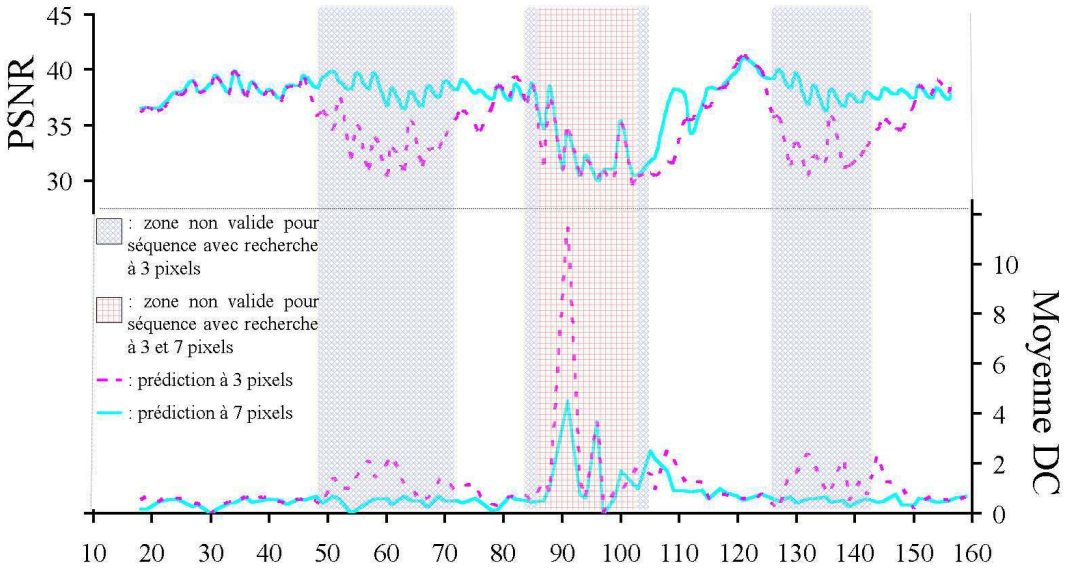
Comparaisons des résultats donnés par les deux méthodes : nous remarquons visuellement que l'estimation donnée par notre méthode est efficace sur les deux types de séquences, excepté entre les images 87 et 101 pour le flux avec recherche à sept pixels (ce qui est normal car le mouvement est plus grand) et sur trois intervalles (de 49 à 71, de 85 à 103 et de 124 à 143) pour le flux avec recherche à trois pixels. Sur ces intervalles, les résultats donnés par notre méthode rapide sont complètement faux (et bien sûr pour chaque mouvement).

Problématique : tout d'abord, évaluons ce qui se passe dans cette partie de séquence. Pouvons-nous résoudre le problème et sinon, comment savoir que nous sommes dans ce cas là ? Sur ces portions de séquences, le mouvement T_x devient supérieur au seuil de recherche (avec un *extremum* à quasiment -12). Nous avons vu dans le paragraphe 1.2 page 49, que le mouvement maximal pour la recherche du macrobloc équivaut, par interpolation, à un mouvement de 7 pixels dans l'image précédente (dans le cas général *IBBPB...*) ; ce qui implique qu'il ne peut y avoir de prédiction supérieure à cette valeur ; celle-ci peut même être plus faible, par décision prise lors du codage. Les mouvements qui sont donnés dans le flux $\mathcal{M}$PEG1-2

pour cette partie ne sont pas les vrais mouvements et par voie de conséquence, le mouvement estimé pour la caméra n'est pas correct. En dehors de ces intervalles, le résultat est proche de l'autre méthode au demi-pixel près. Cette imprécision est normale car la prédiction de mouvement est effectuée sur des blocs beaucoup plus grands (16×16) et au demi-pixel près alors que dans la méthode plus précise, on travaille au tiers de pixel sur des blocs 4×4.

Dans les zones qui posent problème, il faut trouver un moyen, non pour trouver le mouvement directement dans le flux (ce qui n'est pas possible, sans refaire nous-même l'estimation avec des bornes plus grandes), mais pour savoir à quel endroit le résultat est pertinent et où il ne l'est plus, et ce, avec une grande certitude. Pour cela, nous utilisons le poids de l'image d'erreur.

Une solution : en effet, plus la prédiction de mouvement est proche de la réalité, moins l'image d'erreur sera lourde ; *a contrario*, plus son poids est important, ou plus il y a de macroblocs codés au format *I*, moins la qualité de la prédiction sera bonne. Ce résultat découle du fait que l'estimation du mouvement permet, avec une bonne estimation, de minimiser le poids de l'image d'erreur. Pour ce faire, nous utilisons la moyenne des valeurs absolues *DC* des *DCT* des macroblocs d'erreur (FIG. 2.5).



haut - *PSNR* entre les images originales et les images provenant de la décompression des deux flux $\mathcal{M}$PEG1-2 ;
bas - Affichage de la moyenne des valeurs absolues *DC* des images d'erreur (avec deux flux $\mathcal{M}$PEG1-2 : prédiction à 3 pixels et à 7 pixels).

FIG. 2.5 : Courbes de la séquence Stefan Edberg.

Cette figure nous montre qu'il y a bien une augmentation à l'endroit du mouvement supérieur aux bornes de prédiction. De ce fait, l'étude de cette courbe nous permet d'évaluer la pertinence de l'estimation du mouvement de la caméra.

Si nous regardons la courbe de la moyenne des valeurs absolues *DC* des blocs d'erreur (FIG. 2.5-bas page précédente), nous remarquons que cette valeur est très proche de zéro lorsque l'estimation du mouvement est proche de l'estimation donnée par la méthode de référence. Par contre, elle est plus grande (forte augmentation) dans le cas contraire.

Pour cet exemple, regardons les valeurs chiffrées de cette courbe. Dans les zones où notre estimation est proche de l'estimation de référence, nous arrivons à une moyenne des valeurs absolues *DC* sur l'intervalle de 0,45 (0,44 entre les images de 17 à 85 avec une recherche à sept pixels et 0,47 entre les images 17 à 49 pour la recherche à trois pixels). Par contre, pour les zones où le mouvement donné n'est pas pertinent, nous arrivons à une moyenne des valeurs absolues *DC* sur l'intervalle de 1,29 entre les images 51 à 75, de 2,13 pour les images comprises entre 87 et 110 pour la recherche à 3 pixels. Pour la recherche à 7 pixel, nous arrivons à 1,48 entre les images 87 et 110.

Afin d'automatiser le rejet des zones où la prédiction est non valide, il faut arriver à noter les zones de fortes augmentations. Pour cela, si nous normalisons par la valeur maximale de la courbes, la courbe varira entre la valeur zéro et la valeur un. Nous ne garderons que les endroits où la courbe est inférieur à 0,2.

Nous pouvons remarquer (FIG. 2.5-haut) que le *PSNR* est plus élevé aux endroits où l'estimation du mouvement est meilleure. Travaillant à débit de consigne constant, nous ne pouvons rien dire sur le nombre de blocs de type Intra qui se trouvent dans les images d'erreur. En effet, si le programme peut placer des Intras, tout en gardant le bon débit de consigne, il le fera. Il a comme première exigence le débit auquel il essaie de coller au plus près. Si ce dernier est vraiment trop faible, ce qui rendrait au décodage une qualité inacceptable, le programme nous le signale et trouve un débit minimal permettant une visualisation 'correcte' lors du décodage. La deuxième consigne est la qualité.

Pour obtenir le nombre de macroblocs Intra nécessaires dans les images d'erreur afin d'avoir le bon *PSNR*, il faudrait un codeur avec un *PSNR* de consigne. Un tel codeur n'étant pas encore disponible, sa réalisation compléterait de façon pertinente l'objet de cette étude.

Sur la séquence *Flower-Garden*

Nous appliquons notre méthode à la séquence *Flower-Garden* (séquence de 250 images de dimensions 352×288 pixels). Nous l'avons choisie pour son mouvement de translation rapide avec une partie de l'image fortement texturée. Nous utilisons un flux $\mathcal{M}$PEG1-2 codé avec le graticiel du $\mathcal{M}$PEG *Simulation Group 1994* avec un débit de consigne de 1,1 Mbits/s et une recherche revenant à une prédiction de mouvement limitée à sept pixels entre deux images.

Nous obtenons une bonne prédiction du *Pan* à $\pm 0,5$ pixel. Pour les autres mouvements, l'erreur maximale en pixel est toujours inférieure à 1 sur les bords de l'image (*cf.* les courbes dans l'annexe D à la page 169).

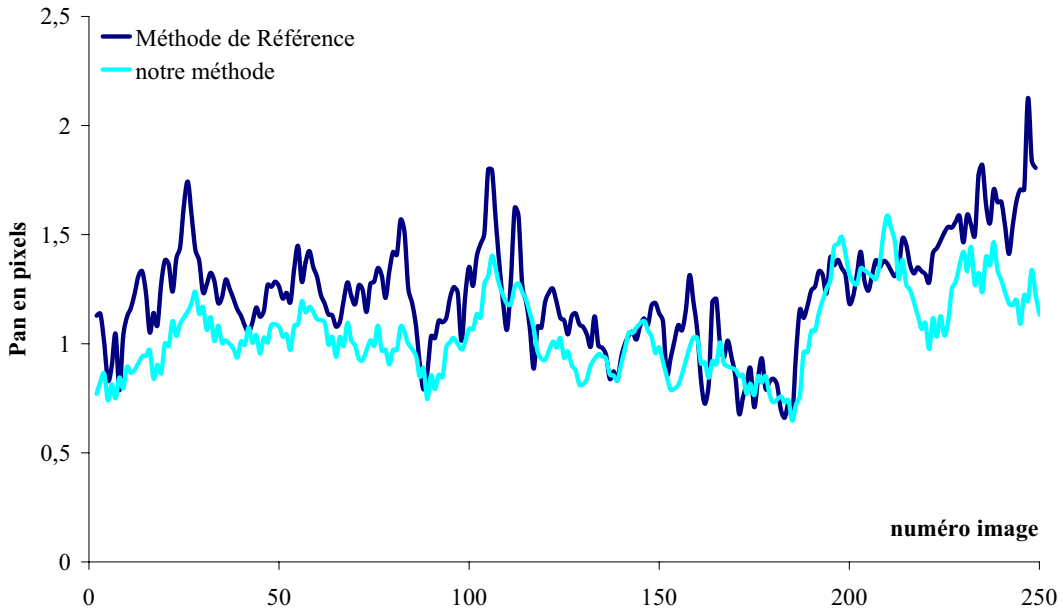


FIG. 2.6 : Comparaison entre notre méthode et une méthode de référence : estimation de T_x

Sur la séquence Bus[2] du groupe ISO/IEC JTC1/SC29/WG11

Nous appliquons notre méthode à la séquence Bus (séquence de 150 images de dimensions 352×288 pixels). Nous l'avons choisie pour son mouvement complexe, deux forts zooms, un mouvement de translation et des objets en mouvement, dont le bus, qui par moment représentent presque la moitié de l'image. De plus, il y a un premier plan et un arrière plan. Il en résulte que cette séquence est contraire à plusieurs hypothèses comme, par exemple, toute la scène se trouve à l'infini et est plane (perpendiculaire à l'axe optique) et les objets en mouvement sont minoritaires. Nous utilisons un flux $\mathcal{M}$PEG1-2 codé avec le graticiel du $\mathcal{M}$PEG *Simulation Group 1994* avec un débit de consigne de 0,9 Mbits/s et une recherche revenant à une prédiction de mouvement limitée à sept pixels entre deux images.

Voici les résultats :

[2] À l'adresse http ://megaera.ee.nctu.edu.tw/Test_Seq/ , il est possible de télécharger le fichier bus_cif.yuv . Les adresses *Web* disparaissant aussi rapidement qu'elles apparaissent, nous ne pouvons dire lorsque vous essaierez si le site existe encore... Erreur 404

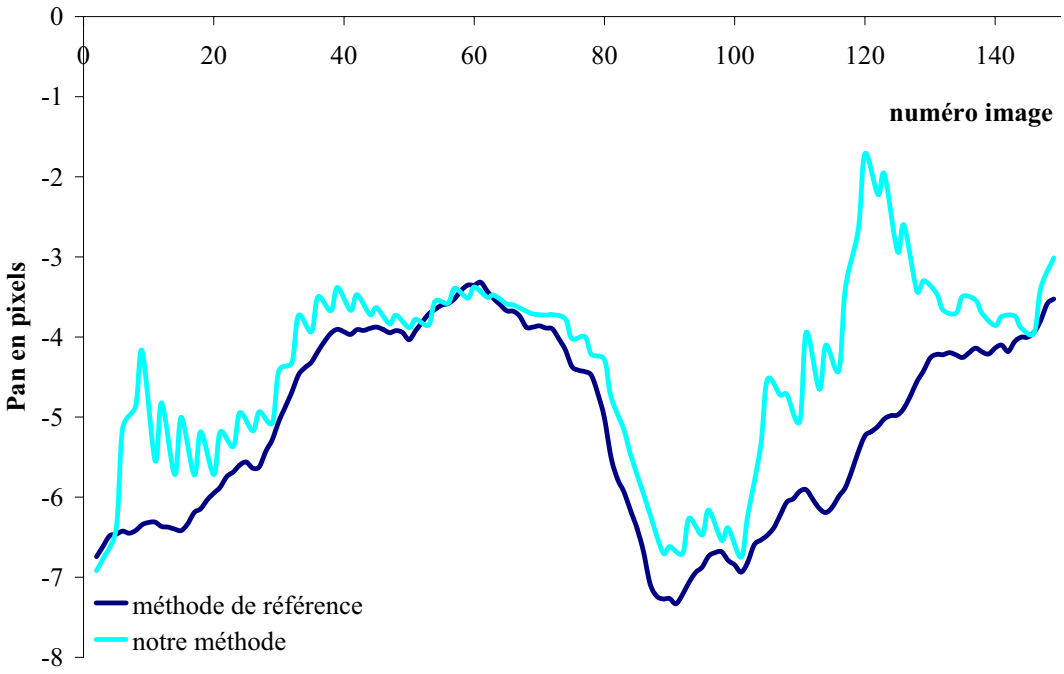


FIG. 2.7 : Comparaison entre notre méthode et une méthode de référence : estimation de T_x

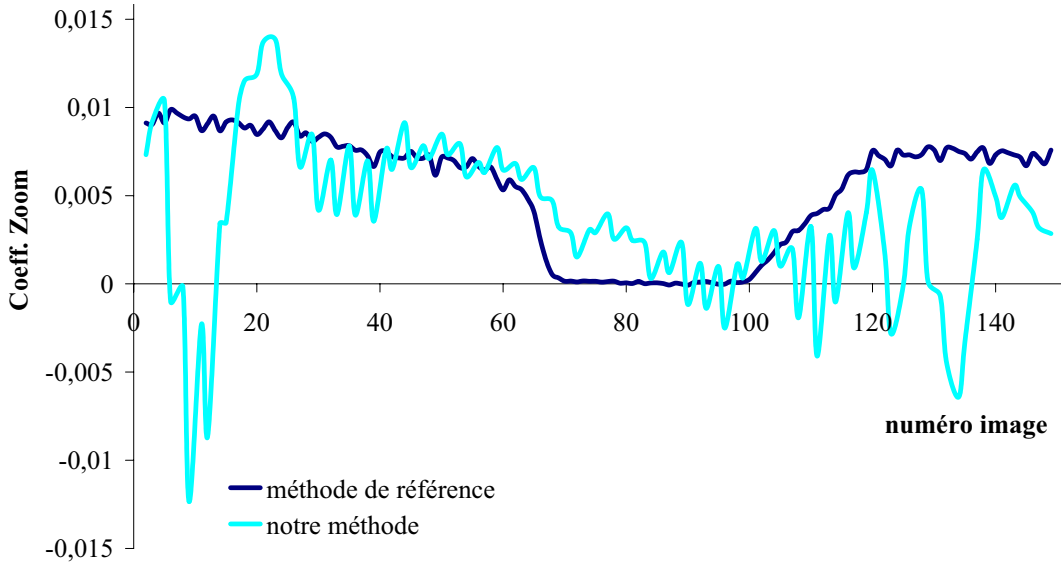


FIG. 2.8 : Comparaison entre notre méthode et une méthode de référence : estimation de T_Z

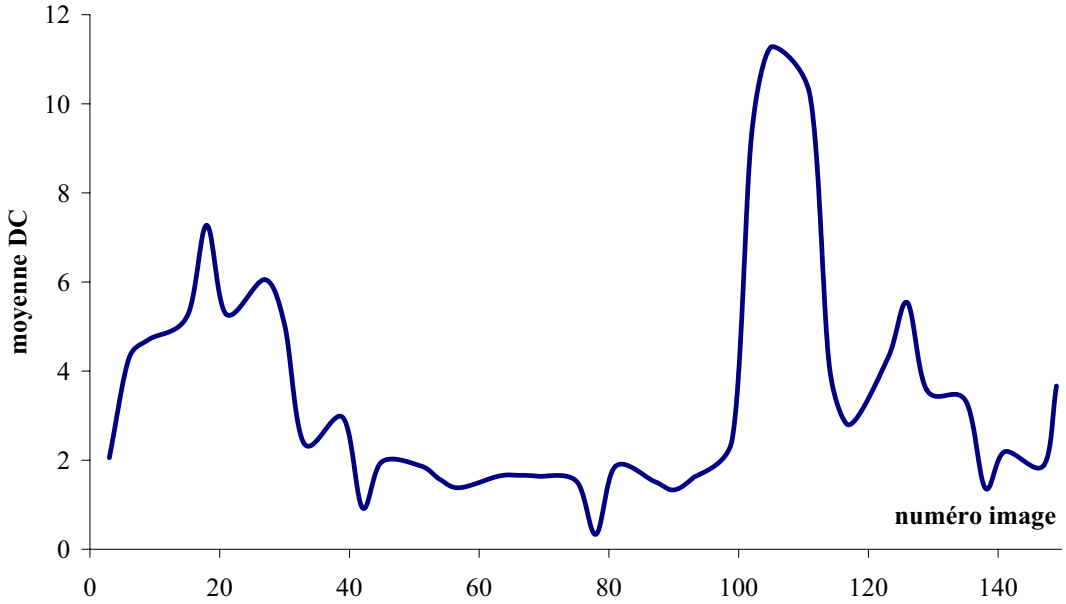


FIG. 2.9 : Affichage de la moyenne des valeurs absolues *DC* des images d'erreur.

Entre les images 40 et 100, nous obtenons une bonne estimation du mouvement (par rapport à la méthode de référence), c'est ce que nous confirme l'étude de la moyenne des valeurs absolues *DC* des blocs d'erreur.

Explications des zones de rejets par rapport au contenu visuel.

Sur les premières images, nous avons un fort zoom et et un fort *pan* qui occasionnent un déplacement supérieur à sept pixels (limite de la recherche entre deux images). En effet, avec un zoom de 0.01, nous obtenons un déplacement sur les bords qui peut atteindre 1,8 pixels, auquel est ajouté un *pan* de 7 pixels. De plus, nous avons plusieurs plans dans notre image, ce qui est contraire à notre hypothèse qui place tous les objets à l'infini. Et pour finir, nous avons des objets qui suivent le mouvement du fond (des voitures) qui ont de très grandes zones d'aplats et qui sont au premier plan.

Après l'image 100, à nouveau un très fort zoom avec des objets en mouvement qui ont une surface supérieure à celle du fond. De plus, à partir de l'image 130, nous avons la route qui apparaît (sa surface va augmenter jusqu'à devenir majoritaire) et qui en plus de former une zone d'aplat est parallèle à l'axe optique, ce qui est aussi contraire à nos hypothèses.

Vous trouverez dans l'annexe D à la page 171 les autres résultats graphiques concernant cette séquence.

Dans l'annexe page 172, vous trouverez également un autre exemple d'estimation du mouvement de la caméra, cette fois-ci pour une séquence de synthèse.

2.3 Conclusion

Dans ce chapitre, nous avons pu mettre en œuvre l'estimation du mouvement de la caméra à partir de sa modélisation vue au chapitre précédent. Pour cela, nous avons utilisé la notion de vecteurs *forward* normalisés introduite dans l'article de M.V. Srinivasan *et al.* [102] en l'étendant aux images de type *B*. L'utilisation de l'algorithme itératif afin d'enlever les vecteurs en dehors d'un intervalle de confiance nous permet d'effectuer cette estimation en quasi temps réel à partir d'un flux $\mathcal{M}$PEG1-2 non totalement décompressé.

Afin d'évaluer la pertinence du résultat ainsi obtenu, nous utilisons les valeurs *DC* de la *DCT* de l'image d'erreur.

Chapitre 3

Conclusion

Nous avons vu dans cette première partie que les méthodes que nous avons mises en œuvre ont permis d'obtenir, en quasi temps réel, le mouvement de la caméra avec des résultats très proches d'une méthode de référence.

À partir des vecteurs de mouvement donnés dans le flux $\mathcal{M}$PEG1-2, nous avons développé un critère nous permettant d'avoir non pas les vecteurs de mouvement par rapport à la ou les images pivots précédente et/ou suivante mais les vecteurs relatifs à l'image précédente (vecteurs *forward* normalisés). Avec ces vecteurs, nous avons pu utiliser le modèle affine simplifié, ce qui nous a permis d'estimer le mouvement de la caméra. Seulement, tous les vecteurs ne sont pas pertinents, car l'optique du codeur n'est pas de faire une prédiction du mouvement, mais de minimiser les données à envoyer. C'est pour cela, que par un traitement itératif, nous enlevons les vecteurs qui sont en dehors d'un intervalle de confiance (nous avons supposé que la distribution des vecteurs se rapprochait d'une distribution gaussienne). L'utilisation d'un intervalle de confiance suppose que la majorité des vecteurs de mouvement répondent aux suppositions que nous avons faites (objets en mouvement minoritaires, fond de l'image à l'infini et perpendiculaire à l'axe optique). Nous notons les vecteurs enlevés lors de l'estimation du mouvement de la caméra, cela nous servira dans la dernière partie de notre manuscrit (extraction des objets en mouvement).

Après tous ces traitements, nous obtenons le mouvement de la caméra. L'étude de l'image d'erreur nous permet d'estimer la pertinence du résultat. Dans le cas défavorable, nous mettrons en œuvre d'autres techniques, beaucoup plus longues mais qui nous donneront le mouvement de la caméra.

Avec tout ceci, pour un plan complet, il est possible de renseigner certains champs $\mathcal{M}$PEG7 concernant le mouvement de la caméra pour une séquence (*cf.* annexe page 146). Ces champs sont sous la forme :

de l'image n à l'image m : le mouvement de la séquence est de ...

Afin de résumer chaque plan (partie ayant le même mouvement, par exemple), il est possible d'extraire une image clef, comme dans l'article de J. Motsch et H. Nicolas [76]. Cette image pouvant servir pour l'indexation sur le contenu.

Perspectives

La première perspective que nous pouvons entrevoir est d'utiliser plus de termes de la *DCT* des images d'erreur. En effet, nous nous sommes pour l'instant cantonnés à la valeur *DC*, mais sans décompresser totalement le flux, nous pouvons utiliser les autres valeurs et ainsi donner une pondération pour chaque vecteur *forward* normalisé afin d'améliorer encore l'estimation du mouvement (opter pour une estimation robuste).

La deuxième perspective est de pouvoir estimer les sept mouvements de la caméra (nous n'en avons pour le moment estimé que quatre : les trois translations et la rotation axiale). Pour cela, il sera nécessaire d'utiliser, ensemble, les vecteurs de mouvement de plusieurs images successives en conservant leur historique. Cela permettra de compenser l'effet de faible ouverture de la caméra, et de ce fait, sur les bords de l'image, de différencier les rotations et le zoom des translations.

Si nous arrivons à estimer les sept mouvements, il sera possible d'effectuer une bonne reconstruction d'une mosaïque comme dans l'article de H. Nicolas ou M. Gastaud et M. Barlaud [79, 43]. Nous appelons mosaïque un assemblage de toutes les images afin d'obtenir une reconstruction du fond de la scène. Il est donc possible de ne transmettre, pour la reconstruction de la séquence, que la mosaïque, le mouvement de la caméra (modèle projectif que nous n'avons pas abordé), les objets et leur mouvement. Avec tous ces renseignements, le fond est recréé pour chaque image de la séquence et l'objet est 'collé' au bon endroit.

Deuxième partie
Segmentation de zones de couleur uniforme en 2D étendue au 2D+t

« Il y a, pour les écrivains français, une qualité plus belle que la couleur : la lumière »
Créer, Édouard Herriot (1872-1957)

Résumé : Nous nous sommes imposé d'utiliser un flux $\mathcal{M}$PEG1-2 ; de ce fait, nous utilisons son espace de couleur qui est Luminance - Chrominances. De plus, nous travaillons dans le flux non complètement décodé, *i.e.* nous travaillons au niveau des blocs et non au niveau des pixels. Avec ces informations, nous segmentons l'image en zones de couleur uniforme. Plusieurs approches sont possibles et en particulier l'approche contour et l'approche région. Pour notre part, nous utilisons l'approche région avec l'introduction d'une distance colorimétrique. Deux blocs (qui peuvent être dans deux images consécutives) seront dans la même zone si leur distance est inférieure à un seuil. De plus, pour améliorer les résultats et comme la nature des contours des zones dans une image est plutôt courbe que droite, nous utilisons les *B-Splines* que nous pouvons faire varier grâce à l'introduction d'une force et obtenir ainsi, des zones de couleur uniforme les plus proches de la réalité (ce qu'un opérateur humain donnerait).

Chapitre 1

La couleur dans le flux $\mathcal{M}$PEG1-2 - données utilisées

1.1 Traitement de la couleur dans $\mathcal{M}$PEG1-2

Un film peut être considéré comme une succession d'images fixes pouvant être codées soit au format RVB, soit, comme pour $\mathcal{J}$PEG, en L-Cr-Cb (vous trouverez dans l'annexe B page 154 une présentation des différents espaces de couleur). Ce passage du RVB au L-Cr-Cb permet d'utiliser la moindre sensibilité de l'œil aux deux chrominances, ce qui autorise un sous-échantillonnage pour ces dernières (FIG. 1.1).

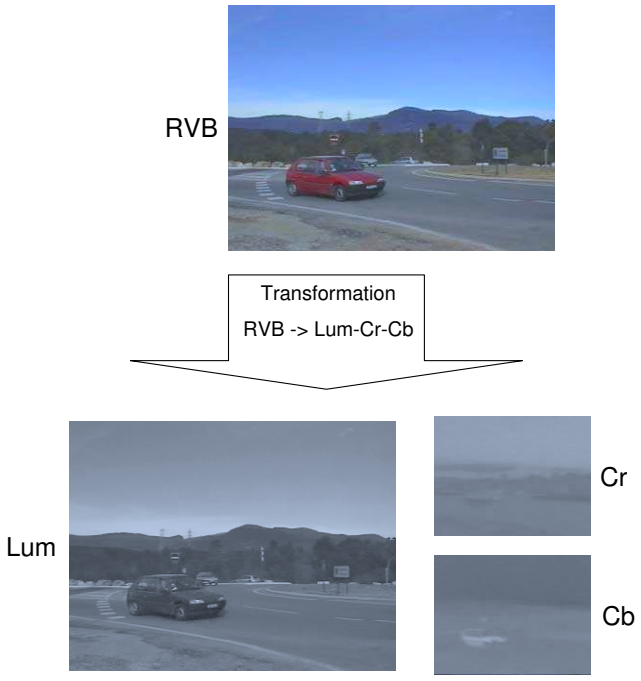


FIG. 1.1 : Décimation sur les chrominances

Cette transformation permet déjà une première compression. Ainsi, si l'image RVB a un poids égal à trois, l'image L aura un poids égal à un, alors que Cr et Cb n'auront qu'un poids de un quart chacune. De ce fait, l'image L-Cr-Cb aura un poids total de seulement 1,5 soit un poids divisé par deux sans perte de qualité visuelle apparente.

MPEG1-2 va mettre en œuvre d'autres algorithmes afin de comprimer le flux de façon performante. Il va découper l'image en bloc de 8x8 pixels tout en sachant que pour la luminance cette taille représente un bloc et pour les chrominances, cela représente un macrobloc décimé (FIG. 1.2).

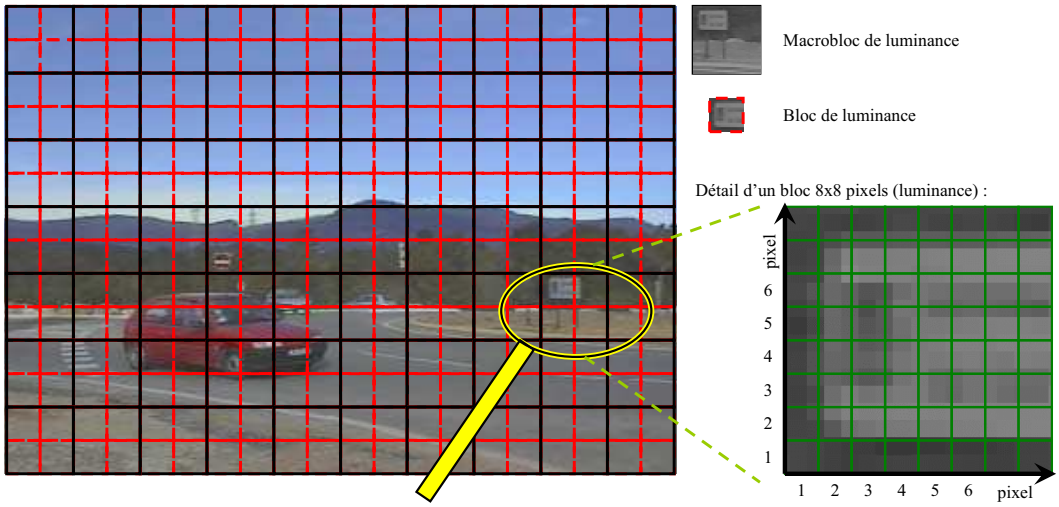


FIG. 1.2 : Découpage de l'image en blocs et macroblocs

Puis, pour chaque bloc (représentant 64 pixels de l'image), une transformation de type *DCT* est appliquée (FIG. 1.3 page ci-contre), ce qui permet d'obtenir des coefficients d'amplitude appliqués à des composantes fréquentielles.

Ces 64 composantes permettent de caractériser tous les motifs qu'il est possible d'obtenir avec les 64 pixels du bloc. Ces coefficients d'amplitude indiquent dans quelles proportions chaque fréquence est présente dans le bloc. La *DCT* donne ainsi une mesure directe de la qualité des détails présents dans ce bloc. La première case dans le coin en haut à gauche (*DC*) contient la valeur moyenne du bloc, *i.e.* la valeur qui apporte le plus d'informations lors de la reconstruction. Plus l'on s'éloigne de cette case, avec le parcours zigzag (FIG. 1.3) plus les motifs rencontrés sont fins ce qui veut dire moins importants pour la reconstruction. L'intérêt de la *DCT* est que généralement les coefficients dans la partie inférieure droite sont proches de zéro, les valeurs significatives étant concentrées sur les premières valeurs lors de la lecture zigzag. L'étude de la *DCT* permet de savoir si la zone est uniforme ou bien texturée. En effet, si la zone est complètement uniforme, la seule valeur non nulle de la *DCT* sera la valeur *DC*.

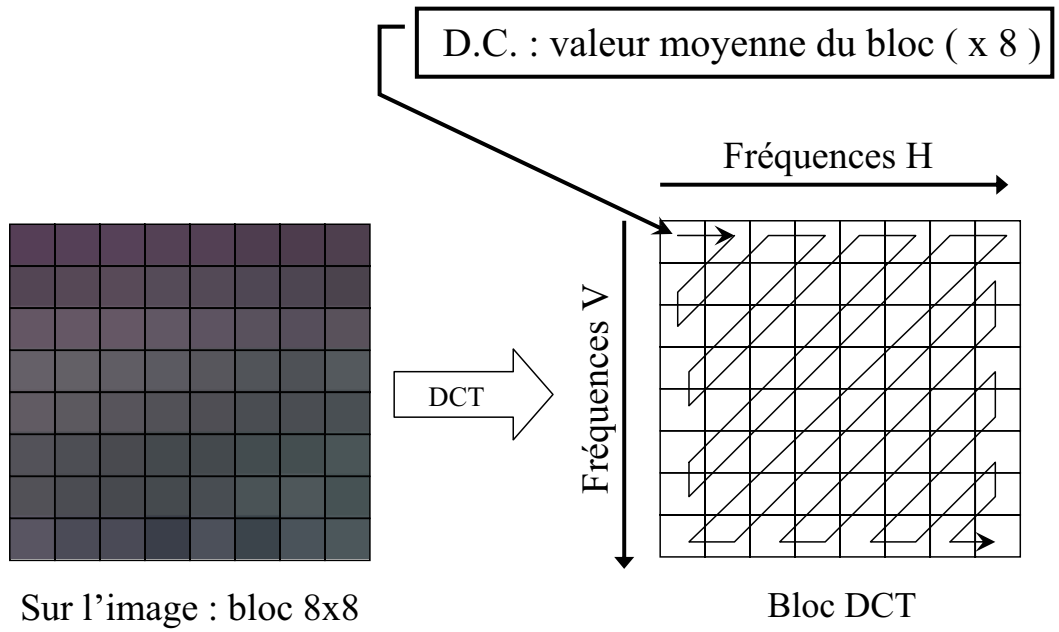


FIG. 1.3 : Représentation de la *DCT* et du parcours zigzag

En résumé, ce codage par *DCT*, en transposant l'image du domaine spatial au domaine fréquentiel, permet d'un côté de classer les informations par ordre d'importance pour la vision humaine, et de l'autre côté, de concentrer l'énergie sur un nombre réduit de coefficients. La *DCT*, en elle-même, ne comprime pas les données, c'est la quantification suivie d'un seuillage, effectué par la suite, qui permettra cette compression (l'annexe A page 143).

Nous avons vu dans le paragraphe 1.2 page 49 le traitement du mouvement dans $\mathcal{M}$PEG1-2 et comment reconstruire la valeur moyenne de chacun des blocs et des macroblocs des images qui ne sont pas codées comme des Intra. Pour les autres parties, nous renvoyons le lecteur à l'annexe A page 145 où il pourra voir l'organigramme de la décompression d'un flux $\mathcal{M}$PEG1-2 (partie vidéo) ; il y trouvera également une explication succincte des méthodes mises en œuvre par $\mathcal{M}$PEG1-2 pour la compression. Ces connaissances ne sont pas nécessaires pour la compréhension de notre exposé mais permettront, à ceux qui le désirent, d'avoir une vision plus globale de la norme.

1.2 Exemple du traitement de la couleur dans $\mathcal{M}$PEG

Dans les deux exemples qui vont suivre, nous utilisons des séquences servant de test pour le *COST* 211[1].

Dans un premier temps, nous regardons à différent débit de consigne, le pic rapport signal sur bruit (*PSNR*) pour chaque type d'images, pour chaque débit et

[1] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*

pour la luminance et les chrominances bleu et rouge (*cf.* l'annexe B page 151 en ce qui concerne les différents espaces de couleur). Le choix du *PSNR* par rapport au *SNR* permet de normaliser le résultat. Nous prenons :

$$PSNR = 10.\log_{10}\frac{255^2}{EQM} \tag{1.1}$$

avec EQM l'erreur quadratique moyenne. Il faut faire atention que le *PSNR* moyen de la séquence (noté ici $\overline{PSNR}$) n'est pas la moyenne des *PSNR* mais pour une séquence de N images avec EQM_i l'erreur quadratique moyenne de l'image i :

$$\overline{PSNR} = 10.\log_{10}\frac{N.255^2}{\sum_{i=0}^{i<N} EQM_i} \tag{1.2}$$

Nous travaillons sur des séquences $\mathcal{M}$PEG1-2 encodées par le graticiel du $\mathcal{M}$PEG *Simulation Group 1994*.

Exemple 1 - Statistiques sur la séquence Stefan Edberg tirée du *COST* 211 : Cette séquence est de dimensions 352×288 pixels et compte 300 images. Elle va être codée avec des débits dont la consigne varie de 0,54 Mégabits à 12,2 Mégabits par seconde pour des *GOP* de 12 images au format *IBBPB...*,, mais aussi pour des *GOP* limités seulement à une image de type *I*.

débit		**26 *I***	**75 *P***	**199 *B***	**moyenne**
$12,2$ **Mb/s**	Lum	$45,06\ dB$	$48,64\ dB$	$47,94\ dB$	$48,02\ dB$
	Cb	$48,64\ dB$	$49,62\ dB$	$49,20\ dB$	$49,42\ dB$
	Cr	$48,64\ dB$	$49,69\ dB$	$49,22\ dB$	$49,47\ dB$
$3,0$ **Mb/s**	Lum	$35,88\ dB$	$34,75\ dB$	$36,71\ dB$	$35,26\ dB$
	Cb	$40,88\ dB$	$38,85\ dB$	$40,03\ dB$	$39,27\ dB$
	Cr	$40,78\ dB$	$38,78\ dB$	$39,97\ dB$	$39,20\ dB$
$1,0$ **Mb/s**	Lum	$29,43\ dB$	$28,08\ dB$	$28,99\ dB$	$28,40\ dB$
	Cb	$36,79\ dB$	$35,25\ dB$	$35,36\ dB$	$35,39\ dB$
	Cr	$36,40\ dB$	$34,74\ dB$	$34,89\ dB$	$34,90\ dB$
$0,54$ **Mb/s**	Lum	$25,79\ dB$	$25,84\ dB$	$25,55\ dB$	$25,76\ dB$
	Cb	$34,41\ dB$	$33,68\ dB$	$33,43\ dB$	$33,68\ dB$
	Cr	$33.90\ dB$	$33.12\ dB$	$32,88\ dB$	$33,12\ dB$

pour des séquences avec des *GOP* de la forme *IBBPB...* de douze images

TAB. 1.1 : *PSNR* moyen pour divers types de débits de consigne avec la séquence Stefan Edberg

Dans le tableau 1.1, nous notons que pour des faibles débits, le *PSNR* moyen est approximativement identique (une différence maximale de 0,3 dB entre les images de type *P* et *B*) alors que pour un fort débit de consigne, la différence de *PSNR* est de plus de 3 dB.

De plus, nous notons que le *PSNR* moyen des chrominances est toujours supérieur à celui de la luminance et que sa progression est plus faible. Cette différence est due au fait que la dynamique des chrominances est moins importante mais aussi étant déjà décimées dans les deux axes, il ne faut pas encore réduire leurs qualités afin de ne pas altérer d'avantage l'image reconstruite.

La comparaison entre les résultats obtenus entre le tableau 1.1 page précédente et le tableau 1.2 de la présente page, nous montre que pour un *PSNR* moyen de 25,73 dB pour la luminance et de 33,68 pour les chrominances, il faut un débit de consigne de 0,54 Mb/s dans le cas *IBBPB...* et un débit de 1,2 Mb/s dans le cas du *GOP* qui contient seulement une image de type *I*. Le débit nécessaire est doublé.

débit		300 *I*
$8,0$ Mb/s	Lum	$40,57 \quad dB$
	Cb	$44,29 \quad dB$
	Cr	$44,29 \quad dB$
$3,0$ Mb/s	Lum	$32,05 \quad dB$
	Cb	$38,16 \quad dB$
	Cr	$37,93 \quad dB$
$1,2$ Mb/s	Lum	$25,81 \quad dB$
	Cb	$34,44 \quad dB$
	Cr	$33,93 \quad dB$

pour des séquences avec des *GOP* réduits à une image de type *I*

TAB. 1.2 : *PSNR* moyen pour divers types de débits de consigne avec la séquence Stefan Edberg

Exemple 2 - Statistiques sur la séquence *Foreman* tirée du *COST* 211 :

Nous faisons exactement la même chose que dans l'exemple précédent.

En comparant avec les résultats précédents, nous remarquons qu'à débit de consigne identique, nous avons un *PSNR* plus élevé ; cela implique une qualité plus grande. Si la qualité est meilleure et que l'on ne peut pas envoyer des images d'erreur plus lourdes (débit identique), cela vient d'une meilleure prédiction de mouvement. Connaissant cette séquence et en la comparant avec la précédente, on se rend compte qu'ici le mouvement de la caméra et des objets est très lent sur une très grande partie de la séquence.

À noter : à qualité visuelle identique, nous avons toujours un débit nécessaire plus que doublé si nous ne transmettons que des *I*.

débit		26 *I*	75 *P*	199 *B*	moyenne
8,1 Mb/s	Lum	$45,08\ dB$	$49,10\ dB$	$48,82\ dB$	$48,50\ dB$
	Cb	$49,31\ dB$	$50,05\ dB$	$49,90\ dB$	$49,94\ dB$
	Cr	$50,29\ dB$	$50,61\ dB$	$50,53\ dB$	$50,56\ dB$
3,0 Mb/s	Lum	$40,38\ dB$	$40,32\ dB$	$42,34\ dB$	$40,75\ dB$
	Cb	$45,56\ dB$	$45,25\ dB$	$45,75\ dB$	$45,40\ dB$
	Cr	$46,96\ dB$	$46,52\ dB$	$47,03\ dB$	$46,68\ dB$
1,0 Mb/s	Lum	$35,29\ dB$	$35,21\ dB$	$36,11\ dB$	$35,43\ dB$
	Cb	$42,45\ dB$	$41,87\ dB$	$42,04\ dB$	$41,96\ dB$
	Cr	$43,65\ dB$	$43,00\ dB$	$43,20\ dB$	$43,11\ dB$
0,5 Mb/s	Lum	$32,24\ dB$	$32,36\ dB$	$32,66\ dB$	$32,42\ dB$
	Cb	$40,66\ dB$	$39,99\ dB$	$39,93\ dB$	$40,03\ dB$
	Cr	$41,62\ dB$	$40,91\ dB$	$40,86\ dB$	$40,95\ dB$
0,28 Mb/s	Lum	$29,26\ dB$	$29,50\ dB$	$29,10\ dB$	$29,38\ dB$
	Cb	$39,11\ dB$	$38,70\ dB$	$38,51\ dB$	$38,68\ dB$
	Cr	$39,46\ dB$	$39,08\ dB$	$38,80\ dB$	$39,04\ dB$

pour des séquences avec des *GOP* de la forme *IBBPB*... de douze images

TAB. 1.3 : *PSNR* moyen pour divers types de débits de consigne avec la séquence *Foreman*

débit		300 *I*
3,0 Mb/s	Lum	$35,86\ dB$
	Cb	$42,91\ dB$
	Cr	$44,13\ dB$
1 Mb/s	Lum	$29,89\ dB$
	Cb	$39,52\ dB$
	Cr	$40,10\ dB$

pour des séquences avec des *GOP* réduits à une image de type *I* .

TAB. 1.4 : *PSNR* moyen pour divers types de débits de consigne avec la séquence *Foreman*

En conclusion de ces deux exemples :

- le passage du M$\mathcal{J}$PEG (*GOP* n'ayant que des images de type *I*) au $\mathcal{M}$PEG1-2 'normal' (*GOP* de type *IBBPB*...) permet de diviser par plus de deux le débit nécessaire au codage du film (avec un débit divisé par deux, nous gardons la même qualité).
- selon le contenu de l'émission à coder, le débit nécessaire sera différent. En effet, nous remarquons que pour la séquence sportive, il est nécessaire d'avoir un débit plus que doublé pour obtenir le même *PSNR* moyen que sur la séquence *Foreman*. Cela explique que la bande passante nécessaire (sur le satellite et le câble) pour diffuser du sport soit plus importante que pour un film

(à qualité identique). Un classement par ordre croissant de bande passante est établi : séries Sud-Américaines, autres séries, films, sport. C'est pour cela, sans avoir vu de séries sud-américaines, nous pouvons dire qu'il ne doit pas y avoir beaucoup de mouvements de caméra, que les artistes doivent souvent rester statiques et qu'il n'y a pas trop de changements de plan.

1.3 Données que nous utilisons

Dans ce qui suit, nous appellerons image mosaïque, une image dans l'espace colorimétrique L-Cr-Cb, ayant les mêmes dimensions que l'image réelle mais qui a, au niveau de chaque bloc, une luminance constante et, au niveau de chaque macrobloc, les deux chrominances constantes.

(a) image initiale (b) image mosaïque

FIG. 1.4 : Visualisation de l'image initiale et de son image mosaïque

Nous nous sommes fixés de ne pas décompresser totalement le flux $\mathcal{M}$PEG1-2. Dans la partie concernant le mouvement, afin d'étudier la pertinence de l'estimation du mouvement de la caméra, nous avions utilisé les valeurs *DC* des images d'erreur ; il n'avait donc pas été nécessaire de décompresser totalement le flux, évitant ainsi le calcul de la *DCT* inverse. Nous continuons dans cette optique en travaillant sur une séquence ne comportant que des images mosaïques. Nous regardons maintenant comment obtenir chaque image mosaïque de la séquence mosaïque.

Pour les images de type *I*-mosaïque : ne prenant pas appui sur une image pivot, elles sont décodables immédiatement. De ce fait, nous travaillons avec une image reconstruite uniquement avec les valeurs *DC* de la luminance et des chrominances et nous obtenons directement l'image mosaïque.

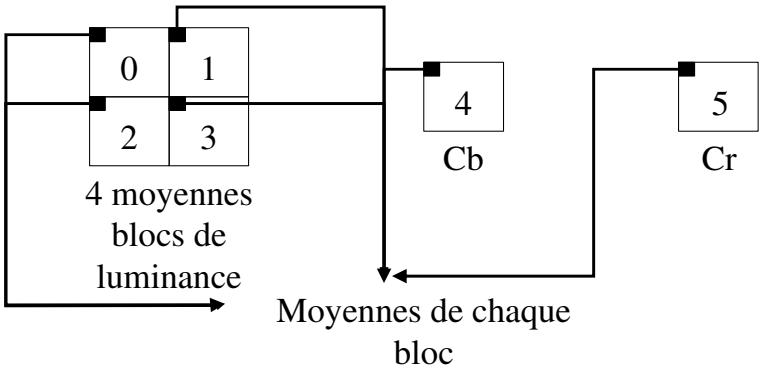


FIG. 1.5 : Reconstruction d'une image de type *I*-mosaïque

Pour les images de type *P*-mosaïque : elles ne sont pas décodables immédiatement, nous devons les reconstruire. Pour cela, nous utilisons l'algorithme de reconstruction des images de type *P* donné par la norme MPEG1-2, avec comme image pivot précédente, l'image mosaïque pivot précédente et comme image d'erreur, l'image mosaïque d'erreur.

Dans la figure ci-dessous, nous utilisons cet algorithme afin de reconstruire un macrobloc de luminance (composé de quatre blocs de luminance) :

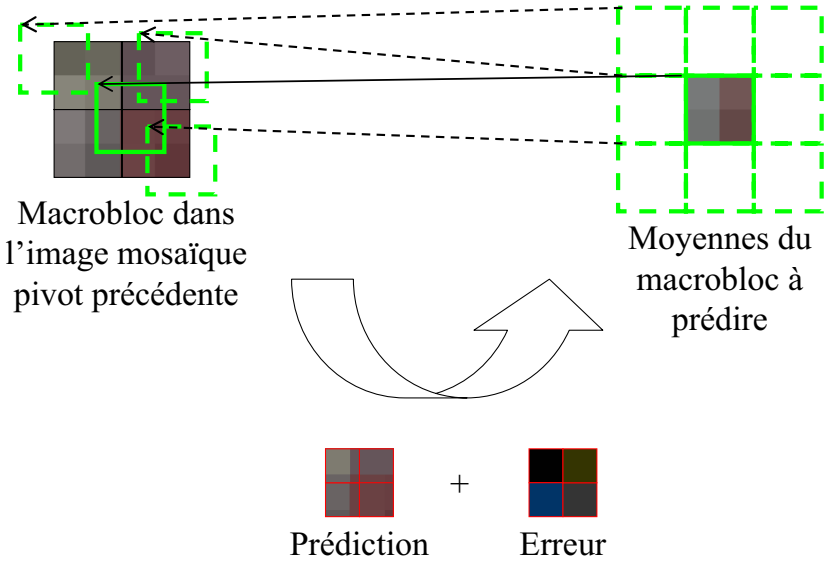


FIG. 1.6 : Reconstruction d'un macrobloc de luminance pour une image de type *P*-mosaïque

Nous utilisons le même *modus operandi* pour les macroblocs de chrominances.

Pour les images de type *B*-mosaïque : de la même manière que pour les images *P*-mosaïque, en utilisant cette fois-ci l'image mosaïque pivot précédente, l'image mosaïque pivot suivante et l'image mosaïque d'erreur.

1.4 Conclusion

Nous avons vu dans ce chapitre comment obtenir les données colorimétriques sur lesquelles nous travaillons. Tout d'abord, comment $\mathcal{M}$PEG1-2 traite la couleur (espace L-Cr-Cb). Dans un deuxième temps, afin de ne pas décompresser totalement le flux, nous avons reconstruit chaque image de la séquence afin d'obtenir une séquence mosaïque (en utilisant uniquement toutes les valeurs *DC* et les vecteurs de mouvement donnés dans le flux).

À partir de là, dans le chapitre suivant, nous regroupons chaque bloc des images mosaïques, afin d'obtenir des zones de couleur uniforme. Pour ce faire, nous posons une distance colorimétrique tout d'abord sur une image mosaïque seule, puis, en utilisant le fait qu'une zone présente sur une image a une forte probabilité d'être présente dans l'image mosaïque d'avant et dans l'image mosaïque d'après, nous utilisons une succession d'images mosaïques.

Chapitre 2

Segmentation en zones de couleur uniforme

2.1 Une définition de la segmentation

La segmentation a pour but de partager une image en régions dont les pixels présentent des caractéristiques semblables. A. Zucher, dans son article [121], définit la segmentation d'images de la façon suivante :

Soient I une image composée de T pixels p_i, P un prédicat défini sur tout sous-ensemble de pixels connexes de I. Segmenter I en régions revient à réaliser une partition de I en N sous-ensembles de pixels $R_1, R_2, \ldots, R_N$, appelés régions, tel que :

- $\bigcup_{n=1}^{N} R_n = I$
- *quel que soit n dans l'intervalle des entiers naturels de 1 à N, quel que soit le couple de points p_1 et p_2 d'un sous-ensemble de pixels R_n, nous pouvons dire qu'il existe un chemin pour relier le point p_1 au point p_2 dans ce même sous-ensemble R_n.*
- *quelque soit n dans l'intervalle naturel de 1 à N, le prédicat $P(R_n)$ est vrai*
- *quel que soit n et m distincts, si R_n est voisin de R_m, cela implique que le prédicat $P(R_n \cup R_m)$ est faux.*

2.2 Les différentes méthodes de segmentation

Les multiples méthodes de segmentation peuvent être classées selon trois dénominations :

1. les algorithmes basés sur la mesure de l'espace, comme par exemple les méthodes de regroupement (*clustering*), ou bien l'utilisation de seuils multiples pour l'étude des histogrammes de couleurs. Nous pouvons citer les travaux de A.M. Bensaid *et al.*, E.C.K. Tsao *et al.*, J.C. Yen *et al.* ou P.Y. Yeng et L.H.

Chen [15, 110, 118, 119] ; dans le domaine compressé, nous pouvons citer les travaux de J. Benois-Pineau et A. Khrennikov [13] ;

2. les algorithmes basés sur les différences entre les pixels comme par exemple la détection de contour avec la possibilité de faire évoluer le contour au cours du temps, ce sont les contours actifs (l'article de A.R. Mansouri *et al.* [67]). Ces contours peuvent être vus de façon variationnelle, nous pouvons citer les travaux fondateurs de M. Kass *et al.* [60], ou bien par des EDP (Équations aux Dérivées Partielles), nous pouvons citer l'article de R. Malladi *et al.* [65] ou bien de V. Caselles *et al.* [21, 22]. Pour notre part, nous baserons notre étude à partir des travaux qui se déroulent au sein de notre équipe : les recherches sur les contours actifs par S. Jehan-Besson *et al.* [54, 53, 55] et les recherches sur les *B-Splines* par F. Precioso *et al.* [90, 89, 91]. À noter, l'approche duale, *i.e.* les similarités des pixels comme par exemple les croissances de régions ou la division-fusion vue dans les articles de S. Zhu et A. Yuille ou de T. Chan et L. Vese [120, 24] ;
3. les algorithmes basés sur des critères physiques, nous pouvons citer : G.J. Klinker *et al.*, B.A. Maxwell et S.A. Shafer ou J.M. Rubin et W.A. Richards [61, 70, 95].

Pour mener à bien la segmentation, après avoir étudié l'approche sur les similarités des pixels, nous utilisons, successivement, deux des approches précédentes. Tout d'abord l'approche numéro un où nous définissons une distance colorimétrique sur une image que nous étendons à une succession d'images. Au sortir de là, nous avons des zones de couleur uniforme. Ayant ces zones, nous avons leurs contours que nous faisons varier (approche des différences entre les pixels) afin d'obtenir une meilleure segmentation.

2.3 Approche détection de régions

Dans cette approche, il est possible de distinguer deux angles de vue :

2.3.1 Méthodes tenant compte de la corrélation spatiale

Un pixel dans une image n'est pas une entité isolée, mais au contraire, il fait partie d'un ensemble qui constitue l'image. Nous pouvons remarquer que les voisins les plus proches d'un pixel donné lui sont habituellement assez similaires. Dans ces méthodes, on utilise aussi bien les caractéristiques colorimétriques des pixels que leurs relations spatiales. Nous pouvons citer quelques méthodes qui utilisent ces relations :

croissance de régions : elles consistent à faire croître des régions en y ajoutant successivement les pixels de leurs pourtours qui satisfont certains critères d'homogénéité. Nous pouvons citer les articles de S. Zhu et A. Yuille

ou S.S. Wong et W.K. Leow [120, 116] qui permettent de grouper d'autorité certaines régions qui n'ont pas une couleur très proche mais qui pourtant appartiennent au même objet ;

par analyse d'un graphe d'adjacence de régions : c'est l'approche utilisée par les champs de Markov et les méthodes de recuit simulé ou *MCM*... Comme dans la thèse de M. Fontaine ou dans l'article de A. Trémeau et N. Borel [39, 109] ;

par fusion-division : nous faisons de façon alternée des divisions puis des fusions. Tout d'abord l'image est divisée en régions homogènes qui respectent des critères globaux, puis les régions adjacentes, qui répondent à des critères locaux, sont fusionnées. On peut soit utiliser les tétra-arbres (*Quadtree*) comme dans l'article fondateur de S.L. Horowitz et S. Pavlidis [49] ou bien dans l'article de V. Coutance [30], soit en utilisant des diagrammes de Voronoï comme dans l'article de J.M. Chassery et M. Melkemi [25].

2.3.2 Les méthodes considérant le pixel comme indépendant

Dans ces méthodes il n'est pas tenu compte du voisinage des pixels, c'est-à-dire qu'elles considèrent les pixels comme des entités indépendantes les unes des autres. Elles sont fondées sur la classification automatique de données multidimensionnelles, considérant les pixels d'une image couleur comme un ensemble non ordonné de vecteurs couleur, qu'il est possible de regrouper dans différentes classes. Dans ces méthodes interviennent deux processus distincts. Tout d'abord l'apprentissage (comme par exemple l'apprentissage compétitif de l'article de T. Uchiyama et M.A. Arbib [111]), consistant à construire des classes de pixels, chaque classe correspondant à une ou plusieurs régions. Après cette opération, les caractéristiques de chaque classe sont déterminées. Ce n'est qu'à partir de là qu'intervient le second processus, la décision d'attribuer, selon ses caractéristiques colorimétriques, une classe à chaque pixel.

Deux familles distinctes mettent en œuvre ces méthodes de classification :

les méthodes supervisées : ce sont des méthodes qui nécessitent une intervention humaine pour la construction des classes ;

les méthodes non supervisées : ce sont des méthodes pour lesquelles la construction des classes est automatique et s'applique à un ensemble d'apprentissage ou à l'image à segmenter.

2.4 Création d'une distance colorimétrique

Afin de réunir des blocs entre eux, il est nécessaire de définir une distance sur les couleurs (luminance et chrominances) entre blocs voisins. Dans un premier temps, nous définissons une distance colorimétrique sur une même image que nous appelons distance colorimétrique 2D. Dans un deuxième temps, nous généralisons cette

distance à la dimension temporelle (distance colorimétrique entre deux images). Dans les deux cas, il faut définir ce que nous entendons par blocs voisins.

2.4.1 Distance colorimétrique 2D

Ici, nous travaillons sur les images mosaïques dont chaque bloc a une valeur constante (*cf.* la représentation de l'image dans la figure 1.4 page 83).

Tout d'abord définissons la notion de voisinage dans une image. Nous dirons que deux blocs sont voisins s'ils ont un côté commun (4-voisinage, FIG. 2.1).

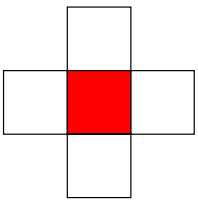


FIG. 2.1 : 4-voisinage pour le bloc représenté en rouge en position (i, j)

Nous nous limitons à ce niveau. En effet, dans toutes nos expérimentations, la recherche au 8-voisinage alourdit inutilement l'estimation pour un résultat quasiment identique. Nous posons la distance entre le bloc rouge et un de ses blocs voisins :

$$dist(f_{i,j}, f_{p,q}) = \frac{\sum\limits_{l \in L} \left[\lambda_l \cdot \left(\frac{f^l_{i,j} - f^l_{p,q}}{moy_l} \right)^2 \right]}{\underbrace{\sum\limits_{l \in L} \lambda_l}_{L=\{lum,Cr,Cb\}}} \tag{2.1}$$

- avec $(p, q) \in \{(i, j-1), (i, j+1), (i-1, j), (i+1, j)\}$, c'est-à-dire un des quatre blocs voisins du bloc en position (i, j) ;
- $f^l_{i,j}$ et $f^l_{p,q}$ représentent la valeur moyenne des deux blocs voisins de type l ;
- moy_l représente la valeur moyenne de l'image mosaïque de type l ;
- λ_l représente un poids fixé empiriquement (qui peut être différent pour la luminance et les chrominances).

Remarque : pour la séparation de deux blocs dans le même macrobloc, λ_{Cr} et λ_{Cb} sont pris nuls. En effet, par la décimation, les macroblocs de chrominances sont codés sur un seul bloc (*cf.* le paragraphe 1.1 page 77 sur le traitement de la couleur par $\mathcal{M}$PEG1-2).

Algorithme mis en œuvre

Avec un seuil de distance colorimétrique S, pour chaque zone de l'image (lors de la première itération, chaque bloc représente une zone), nous regardons les zones qui lui sont voisines. Nous disons que deux zones sont voisines, s'il existe un bloc de l'une qui a un bloc voisin (au sens du 4-voisinage) dans l'autre. Si la distance colorimétrique entre les deux zones est inférieure au seuil S, alors nous disons que ces deux zones n'en forment, en fait, qu'une seule.

Après avoir regardé pour toutes les zones tous les voisins possibles, nous calculons, pour chaque nouvelle zone (réunion d'au moins deux zones dont la distance colorimétrique qui les sépare est inférieure au seuil fixé), sa couleur moyenne (nous faisons ce calcul indépendamment pour la luminance et les chrominances rouge et bleu). Nous remplaçons pour cette zone ses différentes couleurs (car réunion de plusieurs zones) par sa couleur moyenne.

Nous augmentons le seuil S de ε.

Nous itérons notre algorithme jusqu'à un seuil maximal ou un nombre de zones minimal.

Voici l'algorithme en pseudo-code pour une image donnée :

Algorithme 2 Réunion de zones si distance colorimétrique inférieure à un seuil.

```
soit S le seuil pour la distance colorimétrique
soit ε l'augmentation de la distance colorimétrique
soit Ms le seuil maximum de distance colorimétrique
soit Mz le nombre minimum de zones différentes
chaque bloc de l'image représente une zone distincte
tant que (S < Ms) et
|         (nombre de zones différentes > Mz) faire
|   pour toutes zones voisines faire
|   |   si (distance colorimétrique inter zones < S) alors
|   |   |  Mémoriser ces zones comme équivalentes
|   |   finsi
|   finpour
|   marquer toutes les zones comme non traitées
|   pour toutes les zones de l'image faire
|   |   si (cette zone n'a pas été traitée) et
|   |   |   (il existe des zones qui lui sont équivalentes)
|   |   |   alors
|   |   |    calculer la couleur moyenne de ces zones
|   |   |    recoloriser ces zones avec la couleur moyenne
|   |   |    marquer ces zones comme traitées
|   |   |    ne faire plus qu'une zone
|   |   finsi
|   finpour
|   augmenter S de ε
fintant que
```

Déroulons notre algorithme sur un exemple (FIG. 2.2 page ci-contre). Au premier tour de boucle, chaque bloc est une zone indépendante. À l'itération n, nous avons cinq zones. Avec un seuil S, nous pouvons réunir deux couples de zones. Nous obtenons, à la fin de l'itération, seulement trois zones indépendantes. Nous calculons la couleur moyenne sur la luminance et sur les chrominances et c'est avec cette moyenne que nous abordons l'itération n+1 avec un seuil S augmenté.

En premier lieu, nous constatons que cet algorithme s'achève en un temps fini. En effet, si nous augmentons le seuil à chaque tour de boucle, nous agrégeons de plus en plus de zones de couleur. Si le seuil atteint la valeur 255, nous sommes sûrs que tous les blocs voisins auront une distance de couleur inférieure à ce seuil, ce qui induira une image ayant une seule zone de couleur. Ce résultat n'apporte bien sûr aucune information. De même, si nous arrivons à un seuil final trop petit, nous avons une sur-segmentation ; le pire des cas étant celui où chaque zone de couleur aura une surface de un bloc. Là aussi, cela n'apporte aucune information.

De ce fait, il est nécessaire de trouver tout d'abord un seuil minimum, puis une augmentation cohérente et enfin un test d'arrêt. Ces trois paramètres devant permettre de trouver un résultat pertinent en un minimum de temps.

Seuil minimal et augmentation

Trouver un seuil minimal et une augmentation performante sont des recherches qui doivent être faites de concert. Deux alternatives s'offrent à nous, selon que nous désirions soit avoir une convergence plus lente mais avec un résultat plus précis, soit une convergence plus rapide mais avec une imprécision possible :

- soit nous utilisons pour toutes les séquences un seuil initial très faible (de l'ordre de 10^{-3}) et une augmentation du seuil encore plus faible (de l'ordre de 10^{-4}), dans ce cas là, nous aurons un nombre d'itérations assez important pour arriver au résultat (nous allons voir ci-dessous, que nous avons un critère d'arrêt avec la variance), en contrepartie, au plus nous augmentons ce nombre d'itérations, au plus nous nous éloignons du temps réel ;
- soit nous trouvons empiriquement un seuil et son augmentation et nous les appliquons à toutes les autres séquences en les pondérant par un facteur dépendant de la distribution de l'image, par exemple la moyenne.

Nous avons souvent utilisé le deuxième choix et en quelques itérations (nombre inférieur à cinq), nous arrivons à une bonne segmentation en zones de couleur uniforme.

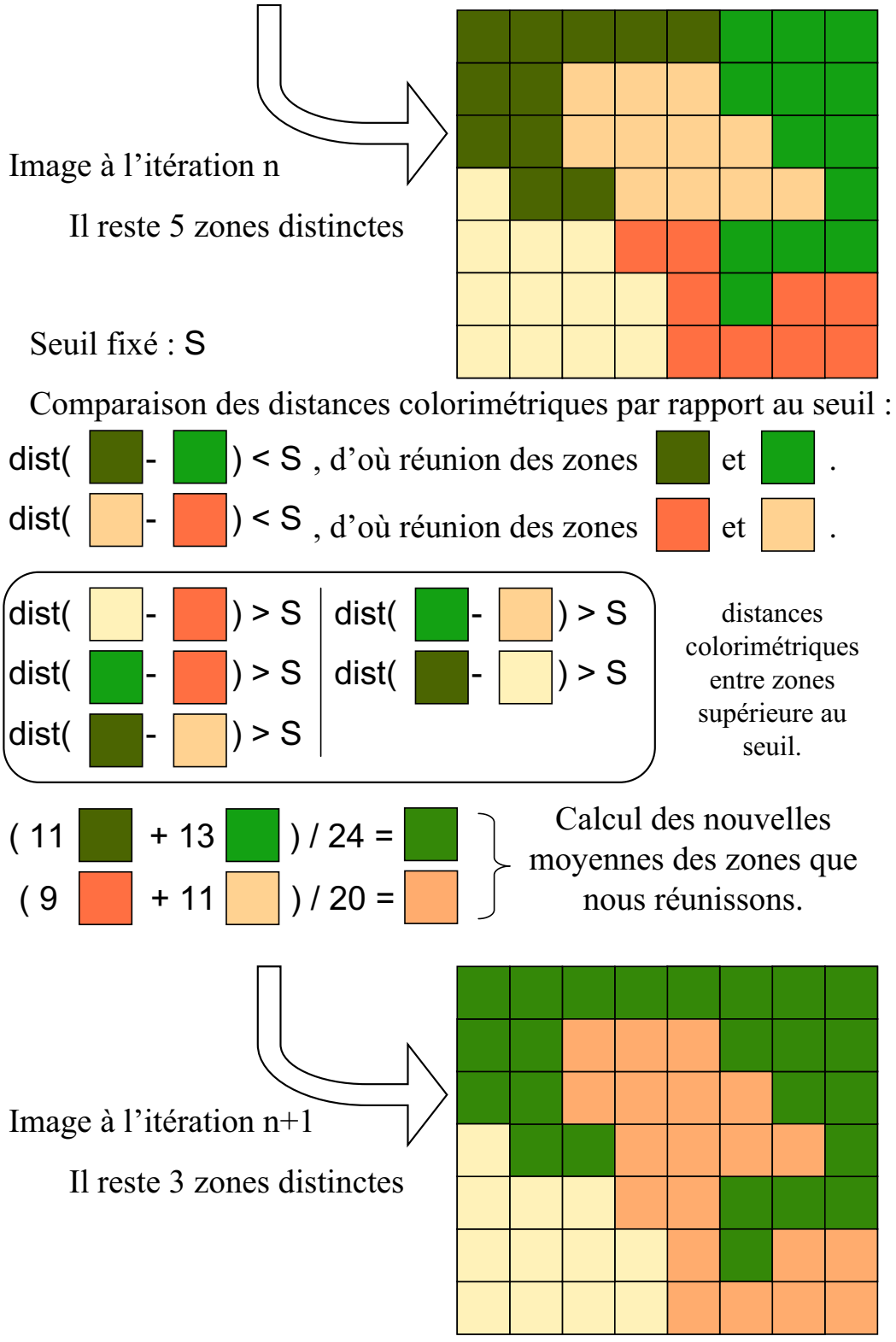


FIG. 2.2 : Algorithme itératif 2D (passage de l'étape n à l'étape $n+1$)

Test d'arrêt

Chaque fois que nous réunissons deux zones de couleur, nous ne comparons pour l'instant que leur distance par rapport à un seuil. Il peut être intéressant de faire intervenir une pondération dans cette réunion. En effet, nous pouvons être dans le cas d'un lent dégradé.

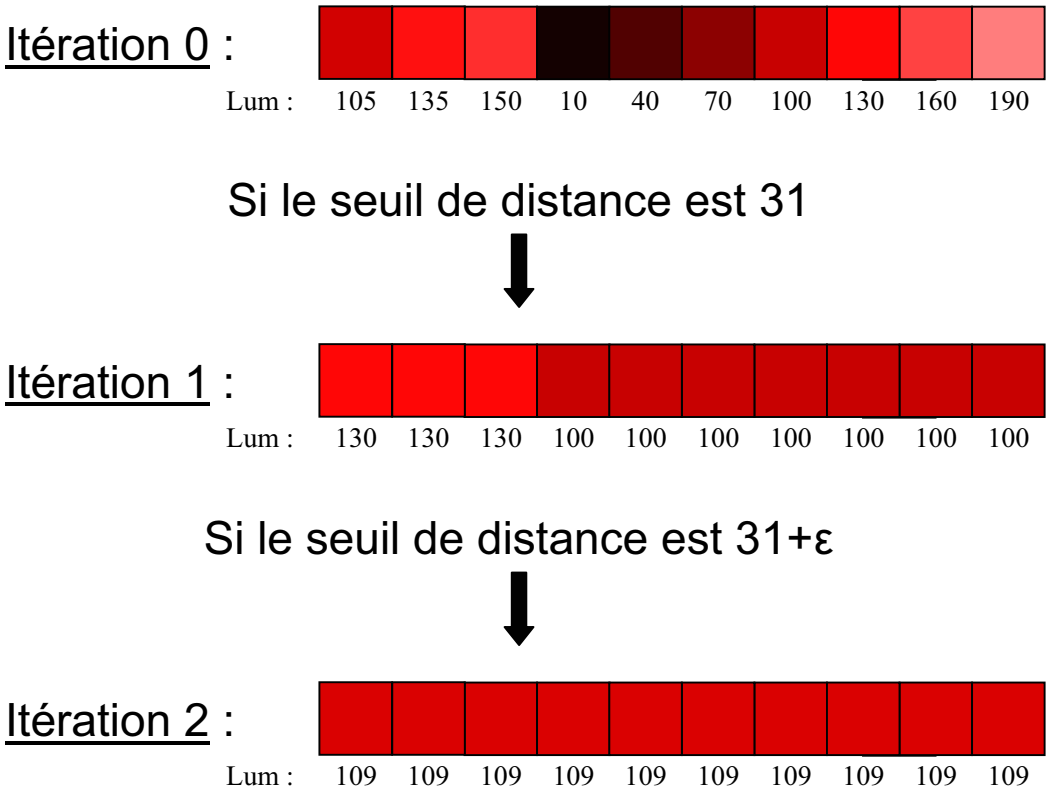


FIG. 2.3 : Cas défavorable du lent dégradé

Dans l'exemple de la figure 2.3, nous illustrons l'intervention de la variance afin de cesser les itérations. Cet exemple est bien sûr un cas d'école, dans la réalité, le seuil final ne devra jamais atteindre une valeur aussi grande car nous risquerions de sous-segmenter et d'avoir des zones de couleur qui englobent des zones pas totalement identiques.

Dans l'exemple donné dans la figure 2.3, nous supposons que le seuil est de 31. À la première itération de notre algorithme, nous obtenons deux zones de couleur uniforme. Si nous poursuivons, nous arrivons à la deuxième itération à un seul objet.

Pour éviter cela, nous faisons intervenir la variance.

La variance de la première zone (à la fin de la première itération) est $\sigma^2 = 525$, la variance de la deuxième zone, est quant à elle de $\sigma^2 = 4200$. Si nous passons à l'itération suivante, nous avons un seuil ε plus grand, ce qui entraîne une réunion des deux zones. Nous avons dans ce cas là une variance totale de la zone : $\sigma^2 = 2814$. La variance diminue de 33% par rapport à la deuxième zone, mais par contre elle

augmente de 536% par rapport à la première. Cette augmentation doit être le signal d'alarme pour ne pas réunir ces deux zones et arrêter nos itérations à l'étape d'avant. Dans tout ceci, nous supposons que le seuil initial a été bien choisi, nous effecturons donc toujours la première itération.

Il faut donc fixer, pour toutes les zones contenant un certain nombre de blocs, un pourcentage maximum d'augmentation de la variance. Pour les deux zones que nous voulons réunir, il faut calculer la variance des deux zones séparées et la variance de la zone entière. Dans le cas où la plus grande des zones n'augmente pas de plus d'un certain pourcentage, nous pouvons réunir les deux zones afin de n'en faire plus qu'une.

Résultat avec une séquence

Voici les résultats obtenus avec une séquence à caméra fixe. Ici le seuil initial de distance couleur est $S = 0,0015$, l'augmentation de ε est de $0,0005$, le test d'arrêt sur la variance est fixé à une augmentation maximale de 20% pour la plus grande zone.

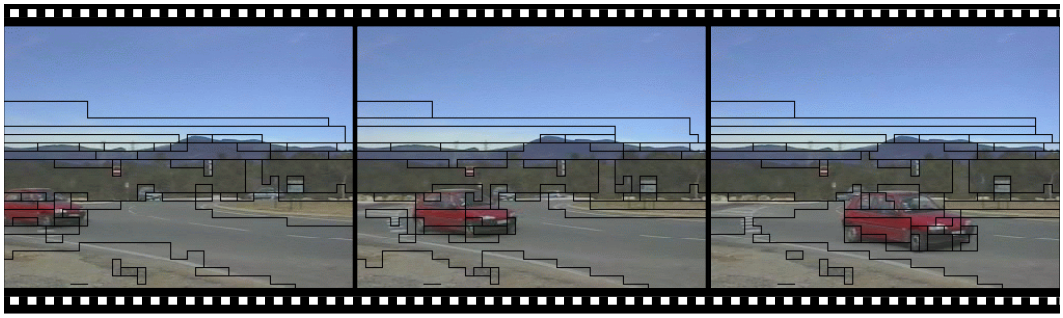


FIG. 2.4 : Résultat de segmentation 2D

Nous allons maintenant utiliser le fait que dans une séquence, une zone de couleur uniforme se trouve généralement dans l'image précédente et dans l'image suivante. Nous étendons notre critère 2D au temps (succession d'images).

2.4.2 Extension 2D+t avec utilisation de la variance

L'ajout au 2D du temps nous fait passer du 4-voisinage au 54-voisinage (FIG. 2.5 page 96). Cela accroît le coût de calcul, mais, par la même occasion, permet d'obtenir une meilleure corrélation temporelle en utilisant ses redondances (meilleure localisation de l'objet dans l'espace). En effet, pour une zone de couleur en mouvement, il y a une forte probabilité qu'elle était déjà dans l'image précédente et qu'elle sera dans l'image d'après, et cela dans une localisation spatiale proche (proche voisinage). Nous verrons par la suite que ce surcoût est contrebalancé par une nette amélioration des résultats.

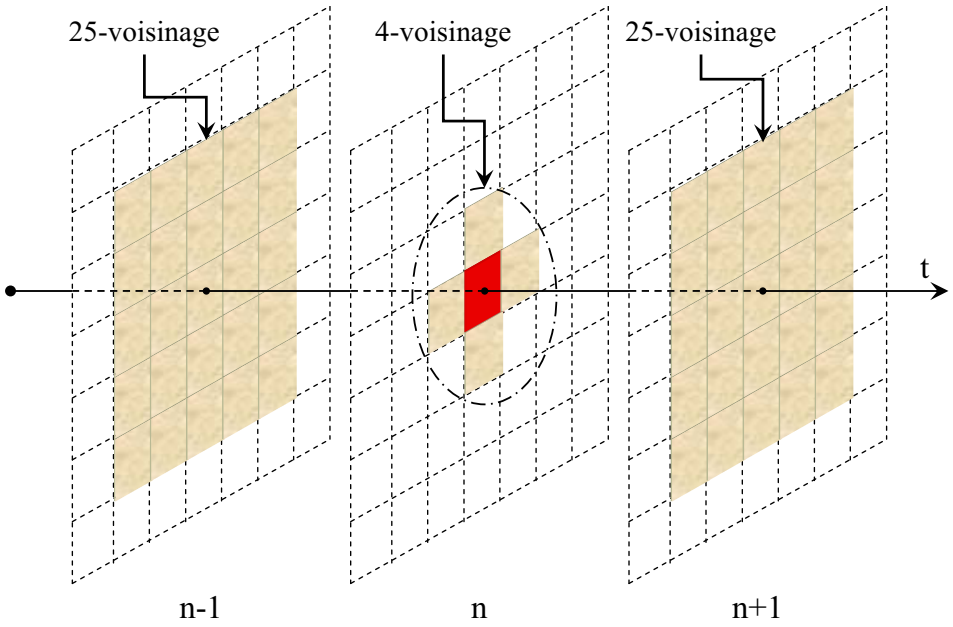


FIG. 2.5 : 54-voisinage à partir du bloc rouge qui est en position spatiale (i, j) dans l'image n (utilisation de trois images successives)

Le fait d'utiliser le 4-voisinage dans l'image en cours a été vu dans le cas 2D. L'utilisation du 25-voisinage dans l'image avant et après autorise un déplacement maximal de la zone de seize pixels. Vu que l'amplitude de déplacement maximal donnée dans le flux est de sept pixels par rapport à l'image précédente, il serait possible de ne considérer que le 9-voisinage dans les images précédente et suivante. Seulement, si nous voulons suivre un objet au cours du temps, qui peut se déplacer de plus de sept pixels entre deux images, nous avons considéré le 25-voisinage qui est un compromis entre le temps de calcul et la qualité de suivi de la zone au cours du temps. Pour l'instant la recherche colorimétrique au cours du temps se fait sans utiliser les vecteurs *forward* normalisés.

Pour le bloc rouge, en position spatiale (i, j), dans l'image n, nous posons une distance colorimétrique entre ce bloc et l'un de ses 54-voisins :

$$dist(f_{i,j,n}, f_{p,q,m}) = \underbrace{\frac{\sum\limits_{l \in L} \left[\lambda_l \cdot \left| \frac{f^l_{i,j,n}}{moy^l_n} - \frac{f^l_{p,q,m}}{moy^l_m} \right| \right]}{\sum\limits_{l \in L} \lambda_l}}_{L=\{lum,Cr,Cb\}} \tag{2.2}$$

– $f^l_{i,j,n}$ représente la valeur moyenne du bloc rouge en position spatiale (i, j), de type l dans l'image n et $f^l_{p,q,m}$ un bloc en position (p, q) qui lui est voisin, de même type l. Ce bloc est dans l'image m avec $m \in \{(n-1), n, (n+1)\}$,

- moy_p^l représente la valeur moyenne de type l dans l'image p,
- λ_l représente une valeur de pondération fixée empiriquement (généralement, cette valeur est différente pour la luminance et les chrominances).

Comme dans le cas 2D, pour la séparation de deux blocs dans le même macrobloc de la même image, λ_{Cr} et λ_{Cb} sont pris égaux à zéro.

Pour chaque bloc (i, j) de l'image n, nous estimons $dist(f_{i,j,n}, f_{p,q,m})$. Si cette valeur est 'très petite', les deux blocs font partie du même objet ; dans le cas contraire, les deux blocs ne font pas partie du même objet. Ici, nous effectuons les mêmes itérations que dans le cas 2D, mais avec cette fois-ci des zones 2D+t. Comme dans le cas 2D, si sur une image nous considérons que deux zones doivent être réunies, c'est alors les deux zones 2D+t qui vont l'être.

Comme dans le cas 2D, lorsque nous réunissons deux zones de couleur qui ont une distance de couleur inférieure au seuil fixé, il peut être intéressant de regarder si cette réunion n'augmente pas trop la variance de la zone. En effet, les deux zones peuvent être proches en moyenne mais avoir une grande disparité dans les blocs (exemple d'un lent dégradé).

Résultat avec une séquence

Voici les résultats obtenus avec une séquence à caméra fixe. Ici le seuil initial de distance couleur est $S = 0,0015$, l'augmentation de ε est de $0,0005$, le test d'arrêt sur la variance est fixé à une augmentation maximale de 20% pour la plus grande zone.

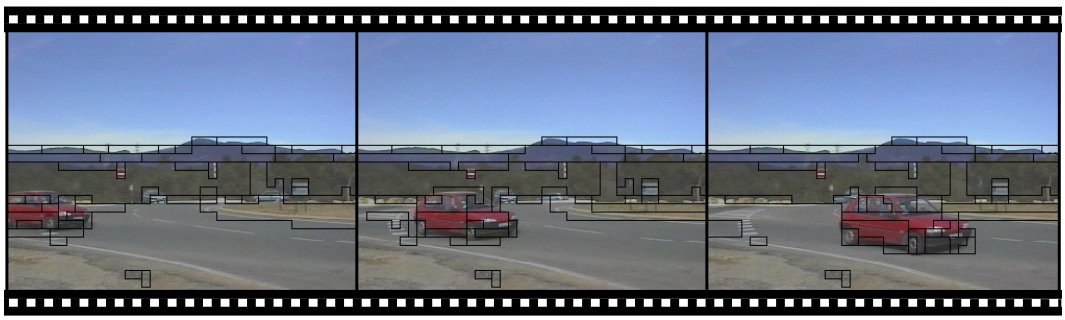


FIG. 2.6 : Résultat de segmentation 2D+t

Nous avons maintenant des zones de couleur uniforme qui sont des agrégats de blocs que nous pouvons suivre d'une image sur l'autre.

Le premier apport du 2D+t par rapport au 2D c'est de pouvoir suivre une zone de couleur uniforme à travers le temps. Cette propriété va nous servir dans la partie suivante, car cela permet de gérer les zones d'occultations et de pouvoir mieux localiser les objets en mouvement. Le deuxième avantage c'est d'avoir moins d'objets. En effet, si nous comparons ce résultat avec le résultat obtenu en 2D pour la même séquence, à la page 95, nous avons maintenant moins de zones dans les endroits de couleur uniforme (ciel, route).

Si nous avons des zones, nous pouvons définir des contours qui les entourent (approche duale : une région est délimitée par une ligne qui représente le contour, et avec un contour fermé nous avons bien une région). Dans la section suivante, nous allons faire varier ces contours afin d'obtenir une segmentation plus proche des objets. Pour cela, regardons en premier lieu la théorie sur les contours actifs, avec deux zones (l'intérieur de l'objet et l'extérieur) et un contour ; puis appliquons-la à notre problématique.

2.5 Les contours actifs

Concernant les contours actifs, nous renvoyons les lecteurs, désirant une étude plus étoffée, à la thèse de S. Jehan-Besson [53] mais aussi aux articles de G. Aubert *et al.* ou S. Jehan-Besson *et al.* [8, 52]. Dans cette partie, nous donnons les grandes lignes de cette technique de segmentation.

Partons de la représentation d'une image (ou domaine) comme dans la figure 2.7 page 98 où nous supposons avoir deux zones, une intérieure et l'autre extérieure.

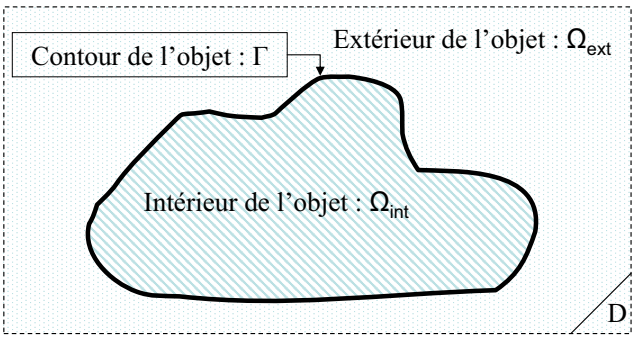


FIG. 2.7 : Extérieur - contour - intérieur

Nous définissons une image comme la réunion de parties toutes distinctes les unes des autres :

$$\Omega_{int} \bigcup \Omega_{ext} = \Omega \text{ et } \Omega_{int} \bigcap \Omega_{ext} = \emptyset \text{ et } \Gamma = \partial\Omega_{int} = -\partial\Omega_{ext}$$

avec Ω_{int} l'intérieur du contour actif, Ω_{ext} l'extérieur de celui-ci et enfin Γ le contour actif.

Deux approches peuvent être faites, soit contours (historiquement la première approche, par exemple l'article de M. Kass *et al.* [60]), soit régions.

Dans l'approche contour, le contour actif évolue à partir d'une équation aux dérivées partielles (EDP) (qui peut être ou non déduite d'une fonctionnelle) incluant uniquement des termes basés contours. En ajoutant des termes basés régions (informations globales sur les régions) aux termes basés contours (informations locales) cela permet de mieux caractériser les régions à segmenter.

Soit :

$$\Gamma(p,\tau) : [a,b] \times [0,T] \rightarrow \mathbb{R}^2 \tag{2.3}$$

une famille de courbes fermées (contours actifs dans le plan de l'image) paramétrées par p qui parcourt le contour (représenté par une équation paramétrique) et τ qui est le paramètre d'évolution de la courbe.

La forme générale de l'EDP régissant l'évolution des contours actifs est la suivante :

$$\frac{\partial \Gamma(p,\tau)}{\partial \tau} = F(p,\tau) \tag{2.4}$$

Pour $\tau = 0$, nous avons le contour initial (défini, par exemple, par l'utilisateur). La force F fait évoluer le contour par déformations successives. La force se décompose en une composante normale et une composante tangentielle, mais d'après C. Epstein et M. Gage ou G. Sapiro [35, 96], si nous nous intéressons uniquement à la géométrie de la déformation et non à sa paramétrisation, alors la force de déformation, que nous notons $\mathcal{F}$, ne dépend que de la partie normale, d'où :

$$\mathcal{F}(p,\tau) = F(p,\tau)\,\mathbf{N}(p,\tau) \tag{2.5}$$

avec $F(p,\tau)$ une force et $\mathbf{N}(p,\tau)$ la normale à la courbe au point p et à l'instant τ. Dans la suite, nous noterons cette normale : $\mathbf{N}$.

2.5.1 Contours actifs

Approche variationnelle

Introduit par Kass *et al.* [60] sous le nom de *snakes*, cette méthode appliquée à une courbe C^2 :

$$\Gamma : [a,b] \rightarrow \mathbb{R}^2$$

est basée sur la minimisation de l'énergie :

$$J(\Gamma) = \underbrace{\alpha . \int_a^b \left|\Gamma'(p)\right|^2 dp + \beta \int_a^b \left|\Gamma''(p)\right|^2 dp}_{\text{énergie interne}} \underbrace{- \lambda \int_a^b \left|\nabla I[\Gamma(p)]\right|^2 dp}_{\text{attache aux données}} \tag{2.6}$$

avec α, β et λ des constantes positives. La partie un de l'équation est utilisée pour imposer une contrainte de régularité sur le contour tandis que la partie deux représente le terme d'attache aux données permettant d'attirer la courbe vers les forts gradients de l'image.

Afin d'éviter la régularisation du second ordre, qui peut occasioner certains problèmes lors de cette minimisation, dont les instabilités numériques, V. Caselles *et al.* [22], posent $\beta = 0$. Ils récrivent la fonctionnelle précédente :

$$J(\Gamma) = \alpha . \int_a^b \left|\Gamma^{'}(p)\right|^2 dp + \lambda \int_a^b g\left(\left|\nabla I\left[\Gamma(p)\right]\right|\right)^2 dp \tag{2.7}$$

avec g une fonction positive strictement décroissante ; ils proposent :

$$g(r) = \frac{1}{1+r^m} \text{ et } m \in \{1,2\}$$

De plus, il montre que minimiser (2.7) revient à minimiser la fonctionnelle suivante :

$$J(\Gamma) = \int_a^b \left\{ g\left(\left|\nabla I\left[\Gamma(p)\right]\right|\right) \left|\Gamma^{'}(p)\right| \right\} dp$$

V. Caselles *et al.* [23] arrivent à l'équation d'évolution du contour actif, appelé contour actif géodésique :

$$\frac{\partial \Gamma}{\partial \tau} = \left[g\left(|\nabla I|\right)\kappa - (\nabla g.\mathbf{N})\right].\mathbf{N}$$

avec κ la courbure de la courbe Γ à l'instant τ et $g\left(|\nabla I|\right)$ une fonction de l'image $g(I)$.

2.5.2 Contours actifs basés régions et utilisation des *B-Splines*

Afin de partitionner une image en deux régions, nous cherchons $(\Gamma, \Omega_{int}, \Omega_{ext})$ qui minimise la fonctionnelle suivante :

$$J(\Gamma) = \int_{\Omega_{int}} k_{int}\left(x, \Omega_{int}\right) dx + \int_{\Omega_{ext}} k_{ext}\left(x, \Omega_{ext}\right) dx + \int_{\Gamma} k_b(x)\, dx \tag{2.8}$$

avec k_{int} le descripteur des objets à segmenter et k_{ext} le descripteur de la région du fond et k_b le descripteur du contour.

Dans la méthode proposée dans l'article de S. Jehan-Besson et M. Barlaud [52], pour une segmentation par contour actif basé région, les auteurs introduisent un schéma dynamique par la méthode des gradients de forme, chaque région dépendant du paramètre d'évolution τ. D'où la dérivée eulérienne de (2.8) par rapport à τ :

$$\begin{aligned} J^{'}(\tau) = & \int_{\Omega_{int}(\tau)} \frac{\partial k_{int}\left(x, \Omega_{int}(\tau)\right)}{\partial \tau} dx \\ & + \int_{\Omega_{ext}(\tau)} \frac{\partial k_{ext}\left(x, \Omega_{ext}(\tau)\right)}{\partial \tau} dx \\ & + \int_{\Gamma(\tau)} \left[k_{ext}\left(x, \Omega_{ext}(\tau)\right) - k_{int}\left(x, \Omega_{int}(\tau)\right) - k_b(x).\kappa(x) + \nabla k_b(x).\mathbf{N}\right](v.\mathbf{N})\, ds \end{aligned} \tag{2.9}$$

avec κ la coubure.

Si $k_b(x)$ est constante, notée par la suite β, nous avons $\nabla k(x).\mathbf{N} = 0$.

Le contour actif $\Gamma(\tau)$ évolue en fonction de τ à partir de $\Gamma(0)$ le contour initial, vers un contour final, avec une force normale au contour $\mathcal{F}.\mathbf{N}$:

$$\begin{cases} \frac{\partial \Gamma(\tau)}{\partial \tau} = \mathcal{F}.\mathbf{N} \\ \Gamma(0) = \Gamma_0 \end{cases}$$

La force $\mathcal{F}$ doit permettre de minimiser l'équation (2.8). Elle doit donc être telle que $J^{'}(\tau)$ soit toujours négative. D'après l'équation (2.9), il suffit de choisir :

$$\mathcal{F} = k_{int}(x, \Omega_{int}) - k_{ext}(x, \Omega_{ext}) + \beta.\kappa(x) + \mathbf{A} \tag{2.10}$$

Le terme $\mathbf{A}$ représente les termes additifs issus des deux premières intégrales de (2.9).

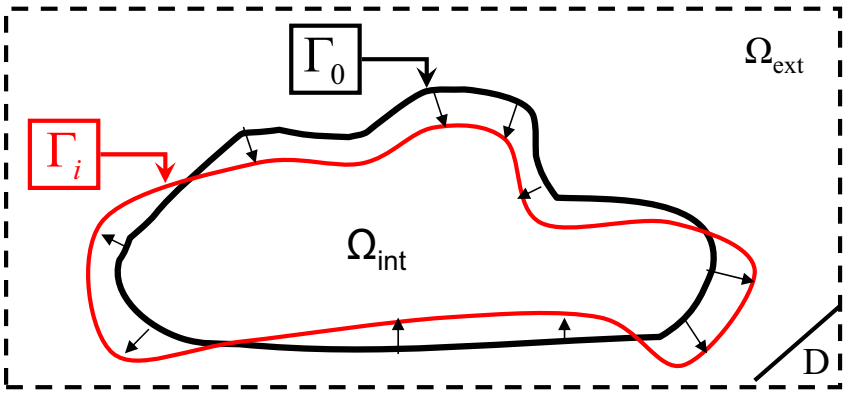


FIG. 2.8 : Variation du contour au cours du temps

Tout d'abord fixons les deux descripteurs régions, $k_{int}(x, \Omega_{int})$ et $k_{ext}(x, \Omega_{ext})$, que nous désirons mettre en œuvre puis dans un deuxième temps estimons le rayon de courbure.

Différents descripteurs régions

Plusieurs descripteurs sont possibles, nous en donnons trois, qui ont été utilisés au sein de notre équipe.

La variance : dans les articles de S. Jehan-Besson *et al.* [52, 53], les auteurs proposent d'utiliser la variance $\sigma^2(\Omega)$:

$$\sigma^2(\Omega) = \frac{1}{\Omega} \int_{\Omega} [I(x) - \mu(\Omega)]^2 \, dx$$

dans ce cas là, si l'on prend comme descripteur région :

$$k(x, \Omega) = \varphi\left[\sigma^2(\Omega)\right]$$

les auteurs arrivent à la force d'évolution selon la normale à la courbe :

$$\begin{aligned}\mathcal{F}(x) = &\varphi\left[\sigma^2(\Omega_{int})\right] + \varphi'\left[\sigma^2(\Omega_{int})\right]\left\{\left[I(x) - \mu(\Omega_{int})\right]^2 - \sigma^2(\Omega_{int})\right\} \\ &- \varphi\left[\sigma^2(\Omega_{ext})\right] - \varphi'\left[\sigma^2(\Omega_{ext})\right]\left\{\left[I(x) - \mu(\Omega_{ext})\right]^2 - \sigma^2(\Omega_{ext})\right\} \\ &+ \beta.\kappa(x)\end{aligned} \tag{2.11}$$

L'entropie : dans l'article de A. Herbulot *et al.* [47], les auteurs proposent d'utiliser l'entropie avec comme descripteur de région :

$$k(x,\Omega) = -q(x,\Omega)\ln\left[q(x,\Omega)\right]$$

avec :

$$q(x,\Omega) = \frac{1}{|\Omega|}\int_{\Omega} K\left[I(x) - I(y)\right]dy$$

et K le noyau gaussien de l'estimation de Parzen.

Les auteurs arrivent à la force d'évolution :

$$\begin{aligned}\mathcal{F}(x) = &k(x,\Omega_{int}) - q(x,\Omega_{int}) \\ &- \frac{1}{|\Omega_{int}|}\left\{E(\Omega_{int}) - 1 + \int_{\Omega_{int}}\left\{K\left[I(y) - I(x)\right]\ln q(x,\Omega_{int})\right\}dy\right\} \\ &- k(x,\Omega_{ext}) + q(x,\Omega_{ext}) \\ &+ \frac{1}{|\Omega_{ext}|}\left\{1 - E(\Omega_{ext}) - \int_{\Omega_{ext}}\left\{K\left[I(y) - I(x)\right]\ln q(x,\Omega_{ext})\right\}dy\right\} \\ &+ \beta.\kappa(x)\end{aligned} \tag{2.12}$$

où :

$$E(\Omega) = \int_{\Omega} k(y,\Omega)\,dy$$

La moyenne : E. Debreuve *et al.* [32] proposent d'utiliser la moyenne $\mu(\Omega)$:

$$\mu(\Omega) = \frac{1}{|\Omega|}\int_{\Omega} I(x)\,dx$$

en prenant comme descripteur région :

$$k(x,\Omega) = \left[I(x) - \mu(\Omega)\right]^2$$

avec $I(x)$ la valeur du pixel au point x et $\mu(\Omega)$ la moyenne colorimétrique de la zone considérée. Dans ce cas là, nous sommes dans le cas où $\mathbf{A}$ est nul (*cf.* thèse de S. Jehan-Besson [53] page 49). De ce fait, la force d'évolution selon la normale à la courbe est :

$$\mathcal{F}(x) = \left[I(x) - \mu(\Omega_{int})\right]^2 - \left[I(x) - \mu(\Omega_{ext})\right]^2 + \beta.\kappa(x) \tag{2.13}$$

Pour notre part, nous prendrons ce dernier descripteur.

Estimation de la courbure

Afin d'utiliser cette force, il nous faut estimer la courbure κ :
soit $M(l) = (x(l), y(l))^t$ le point qui parcourt le contour ; la courbure signée, au point M, sera estimée par l'expression :

$$\kappa(l) = \frac{\ddot{x}(l)\,\dot{y}(l) - \dot{x}(l)\,\ddot{y}(l)}{\left[\dot{x}(l)^2 + \dot{y}(l)^2\right]^{3/2}} \tag{2.14}$$

L'estimation numérique directe de la courbure peut demander beaucoup de ressources système et être imprécise suivant la représentation du contour, c'est pour cela que nous utilisons les *B-splines*. Nous partons du savoir-faire développé au sein de notre équipe à partir des articles de F. Precioso et M. Barlaud [89, 90] et F. Precioso *et al.* [91]. En effet, l'utilisation de *B-splines*, qui utilisent l'interpolation par *cubic B-splines*, permet de modéliser le contour et d'obtenir directement la courbure (pour une vision plus précise des *B-splines*, nous renvoyons le lecteur à l'annexe E page 175).

2.6 Appliquons les *B-splines* à notre problématique

Nous partons de zones de couleur uniforme composées par des blocs de 8x8 pixels. Dans la figure 2.9 page suivante, nous avons une zone de couleur uniforme symbolisée par la surface entourée en rouge. Nous fixons, au milieu des côtés extérieurs de chaque bloc, les points d'échantillonage, ou nœuds, par lesquels va passer la *spline* (ils sont symbolisés par les points noirs). À partir de ces points, nous estimons la *spline* (sur la figure, elle est symbolisée par la courbe bleue).

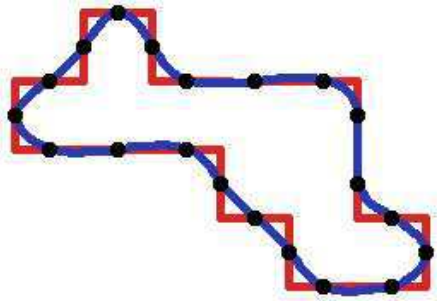


FIG. 2.9 : Représentation de la zone de couleur uniforme composée de blocs (8x8 pixels) par une zone délimitée par une courbe (*B-Spline*) afin de pouvoir lui appliquer des forces.

Nous arrivons pour une image segmentée en zones de couleur uniforme à ce résultat (FIG. 2.10).

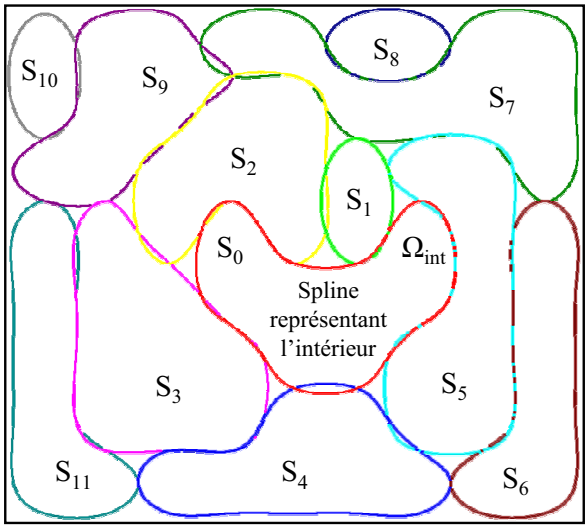


FIG. 2.10 : Exemple d'une image ayant des zones de couleur uniforme séparées par des *splines*.

Nous désirons maintenant faire évoluer le contour de chaque *spline* de façon indépendante. Pour cela, il est facile, pour une *spline* donnée (que nous désirons faire évoluer) de l'étiqueter Ω_{int}. Maintenant, il nous reste à poser Ω_{ext}.

La première possibilité est de supposer que nous sommes dans le cas :

$$\Omega_{int} \bigcup \Omega_{ext} = \Omega \text{ et } \Omega_{int} \bigcap \Omega_{ext} = \emptyset \text{ et } \Gamma = \partial\Omega_{int} = -\partial\Omega_{ext}$$

alors Ω_{ext} est la réunion des intérieurs de toutes les autres *splines* et des zones non étiquetées (celles qui sont entre les *splines*). Seulement nous partons déjà avec des zones de couleur uniforme, nous utilisons donc les zones adjacentes. De plus, nous

négligeons les inter-zones car elles ont une petite surface par rapport à la surface totale des *splines*. Ce qui fait que nous prenons pour Ω_{ext} les zones hachurées en rouge de la figure 2.11.

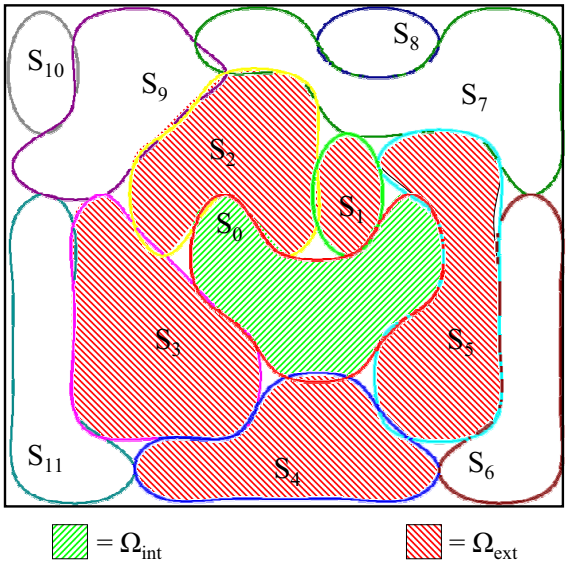


FIG. 2.11 : Intérieur et extérieur pour une des *splines* donnée.

Maintenant reste à estimer la force vue dans l'équation (2.13).
Nous utilisons l'équation (2.13) page 103, et cela pour chaque couleur, d'où :

$$\underbrace{\mathcal{F}(x) = \sum_{l \in L} \left\{ \left[I_l(x) - \mu_l(\Omega_{int}) \right]^2 - \left[I_l(x) - \mu_l(\Omega_{ext}) \right]^2 \right\} + \beta.\kappa}_{L = \{lum, Cr, Cb\}}$$

Cette force s'applique en x, nœud du contour, perpendiculairement au contour (normale vers l'intérieur de l'objet, si Ω_{int} est parcouru dans le sens direct).

En chaque nœud qui définit le passage de la courbe (autre que ceux sur le bord), il passe une autre *spline*. De ce fait, Il va y avoir deux estimations de force pour chaque nœud. Seulement, à la première itération, pour un nœud donné (ceci n'est pas vrai sur les bords) il va y avoir deux forces (une force pour chaque *spline*). Nous n'appliquons pas ces forces indépendamment, sinon nous aurions, dès la première itération, la quasi-totalité des points de contact qui deviendraient distincts. Dans ce cas là, deux problèmes pourraient survenir. Le premier problème est la superposition de zones, ce qui n'est pas possible, car une zone de couleur uniforme ne peut appartenir qu'à une seule zone, sinon cela voudrait dire que les deux zones sont en définitive confondues. Le deuxième problème est que des parties de l'image n'appartiennent à aucune zone de couleur uniforme, cela arrive dans le cas de rétrécissement de zones. Pour éviter tout cela, nous prenons la moyenne des deux forces qui s'appliquent en un point, afin de n'avoir qu'un seul mouvement de celui-ci.

C'est ce que nous illustrons dans la FIG. 2.12 pour le mouvement d'un seul point.

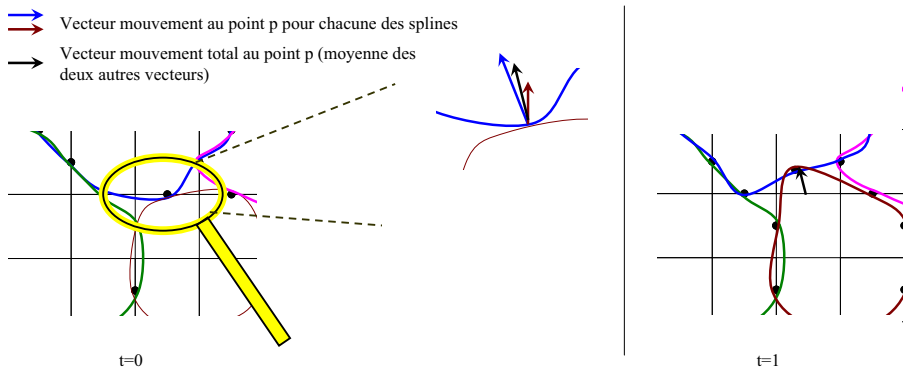
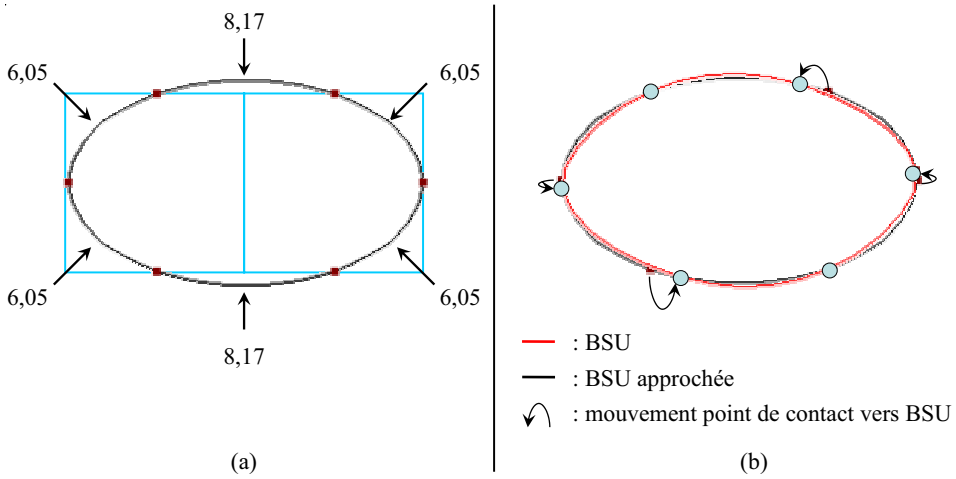


FIG. 2.12 : Variation temporelle en un point de la courbe

À chaque itération, nous estimons la moyenne de la zone et nous réunissons les zones voisines qui ont une différence colorimétrique inférieure à un seuil.

2.6.1 Approximation du cas *B-Splines Uniformes*

Dans tout ce que nous venons de faire, nous sommes partis de l'hypothèse que les points étaient équidistants (même distance curviligne). En effet, nous avons basé notre théorie sur l'usage de *B-Splines Uniformes* appelées *BSU*.



(a) : *spline* avec l'approximation - les valeurs représentent les longueurs des arcs ;
(b) : visualisation de la spline approximée et de la *BSU* - chaque arc fait 6,71.

FIG. 2.13 : Affichage d'une *spline* et validation de l'approximation de la *BSU* dans le cas initial pour deux blocs formant un rectangle.

Or, si au départ cette hypothèse est presque vérifiée (FIG. 2.13) elle le devient de moins en moins. En effet (FIG. 2.13-b), les deux courbes ont approximative-

ment la même forme et en particulier un rayon de courbure semblable. Comme nous utilisons les *splines* pour la rapidité d'estimation des rayons de courbure, nous admettons que nous pouvons utiliser la méthode *BSU*.

2.6.2 Suivi de l'hypothèse *BSU*

Afin de rester dans l'approximation *BSU* , il n'est pas possible de ré-échantillonner les points de chaque *spline* indépendamment, car des points de liaisons seraient supprimés, et comme nous l'avons vu un peu plus haut, sans point de liaison, nous risquons les recouvrements ou les zones non étiquetées. Cette réorganisation doit donc se faire sans rompre les contacts, d'où un intervention locale.

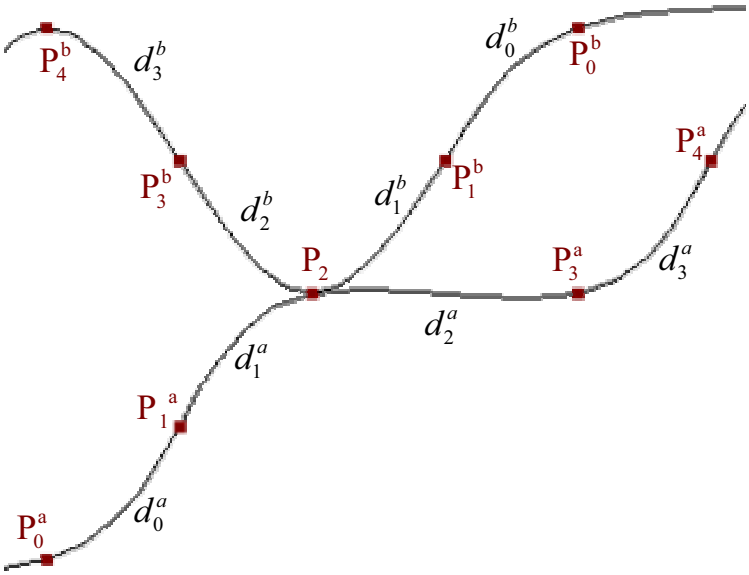


FIG. 2.14 : Point de contact entre deux *splines* - rendre les arcs de courbes de longueurs égales pour chaque *spline*.

Dans la figure 2.14, nous avons deux *splines* qui ont comme point commun le point P_2. Pour la distance (notée d) ou pour le point (noté P), nous notons en exposant le nom de la *spline* qui peut être soit a soit b et en indice le numéro du point ou de la longueur de l'arc.

Posons $S_i(t)$, la i^{eme} équation paramétrique, en fonction de t, de la courbe qui permet de la décrire ou de l'interpoler :

$$S_i(t) = \begin{cases} x_i(t) \\ \\ y_i(t) \end{cases} \quad \forall i \in [0,1]$$

La longueur totale de l'arc de courbe, notée d_i, est égale à l'intégrale de $S_i(t)$ sur l'intervalle $[0,1]$; d'où dans notre cas avec deux *splines* :

$$d_i^l = \int_0^1 \left| \sqrt{\left[\dot{x}_i^l(t)\right]^2 + \left[\dot{y}_i^l(t)\right]^2} \right| dt \quad l \in \{a,b\}\ ,\ i \in \{0,1,2,3\} \tag{2.15}$$

Si nous sommes exactement dans le cas *BSU*, nous avons : $d_0^l = d_1^l = d_2^l = d_3^l$ avec $l \in \{a,b\}$. Mais ce n'est pas exactement notre cas car nous avons bougé le point P_2 en lui appliquant la force calculée précédemment. Il faut maintenant, bouger localement P_2 (qui est le point commun aux deux *splines*) afin de tendre vers le cas *BSU* et nous laissons fixes les autres points. Le fait de le faire bouger localement, implique une modification de ses bornes d'intégration. Nous avons :

$$\begin{cases} d_0^l & = \int_0^1 S_0(t)dt \\ d_1^l(v) & = \int_0^v S_1(t)dt \\ d_1^l(v) & = \int_{v-1}^1 S_2(t)dt \\ d_0^l & = \int_0^1 S_3(t)dt \end{cases} \quad \forall l \in \{a,b\}, \forall v \in]0,1]$$

Afin de trouver la position optimale P_2, nous minimisons le critère :

$$A(v) = \sum_{l \in \{a,b\}} \left\{ \left[d_0^l - d_1^l(v)\right]^2 + \left[d_1^l(v) - d_2^l(v)\right]^2 + \left[d_2^l(v) - d_3^l\right]^2 \right\}$$

Nous cherchons un *extremum* local qui doit être proche de 1 ; pour cela, nous dérivons le critère par rapport à la variable v et nous trouvons la (ou les) valeur qui va (ou vont) l'annuler. Nous prendrons celle qui est la plus proche de 1. Il faut vérifier que c'est bien un *minimum* et non un *maximum*.

Test d'arrêt :

Il y a deux tests d'arrêt qui vont être utilisés pour stopper l'évolution d'une *spline* et de tous ses points de contact avec les autres *splines* voisines :

1. si sur un nœud (qui est commun à deux *splines* sauf sur les bords), la force que nous lui appliquons (composition de deux forces, une pour chaque *spline*) le fait osciller, nous marquons le nœud comme point oscillant et nous arrêtons le calcul des forces lorsqu'une majorité de points sont dans ce cas là. Cette approche pour la détermination des points oscillants est pour l'instant purement empirique et demanderait une solution théorique.
2. si pour une *spline* nous nous éloignons trop de l'approximation *BSU*, nous arrêtons d'appliquer les forces sur cette *spline*.

À chaque fois, nous vérifions si la couleur moyenne de la nouvelle zone déformée n'a pas une distance colorimétrique avec une de ses voisines inférieure au seuil. Dans ce cas là, nous réunissons les deux zones (2D+t) ensemble. De plus, si sa surface devient trop faible, nous la réunissons d'autorité avec une de ses voisines.

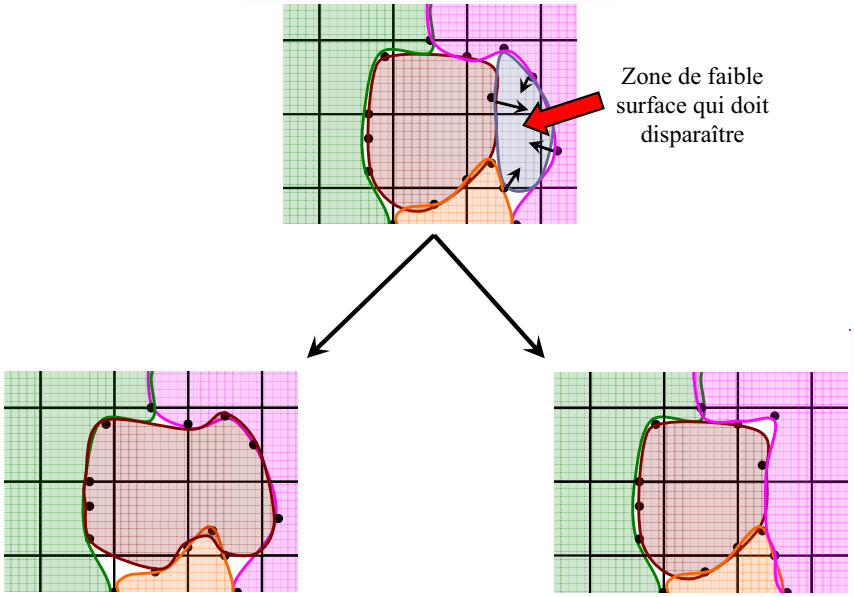


FIG. 2.15 : Suppression d'une zone de couleur uniforme de faible surface en la collant avec une de ses voisines (c'est la distance colorimétrique qui donnera la bonne fusion).

2.7 Résultats

Sur la séquence Voiture

Avec le critère de couleur 2D+t (2.2), nous avons pu segmenter et suivre dans le temps des zones de même couleur. Reprenons la segmentation en zone 2D+t vue sur la séquence voiture dans la figure 2.6 page 97 et remplaçons les segments par des *splines*, nous obtenons la figure suivante :

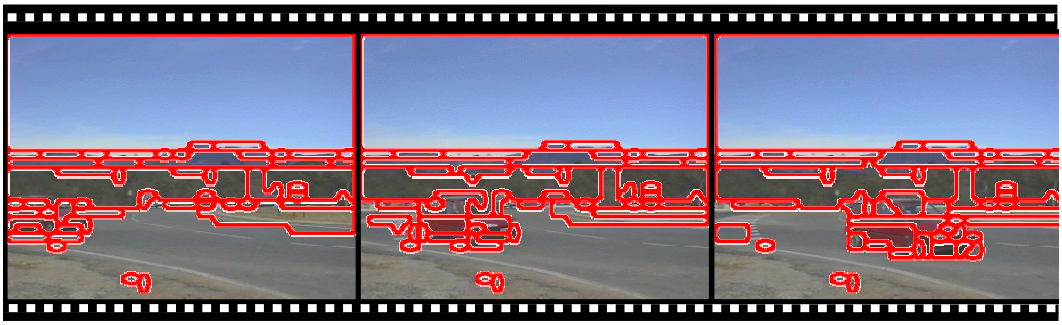


FIG. 2.16 : Images avec zones de couleur uniforme

Sur la séquence Lion

Nous utilisons notre méthode avec la séquence *Lion* du *COST* 211[1]. Les images comportent 288 lignes et 352 colonnes. La figure 2.17 montre le suivi des quatre plus grandes zones découvertes après cinq itérations du critère 2D+t. Nous avons mis autour de l'image 77 le mouvement du centre de gravité de chaque zone, soit sur l'axe x, soit sur l'axe y, et ceci au cours du temps De plus, nous avons mis l'évolution de la surface au cours du temps.

Nous remarquons que moins la zone est texturée, plus elle est grande. Ce qui fait que sur les zones fortement texturées nous obtenons des zones de même couleur réduites à un bloc. Une zone réduite à un bloc est une zone sur-segmentée. Nous regardons si ce bloc est esseulé dans le temps, dans ce cas là, il faut le coller d'autorité avec un objet, plus grand, qui lui est le plus proche (spatialement et colorimétriquement). De plus, sur cette représentation, nous remarquons des zones de surface de un ou deux blocs qui appartiennent à la zone trois mais qui ne sont pas attachés physiquement (sur cette image) à la grande partie ; ceci vient de ce qu'au cours du temps ils ont eu un bloc en commun par le 54-voisinnage récursif.

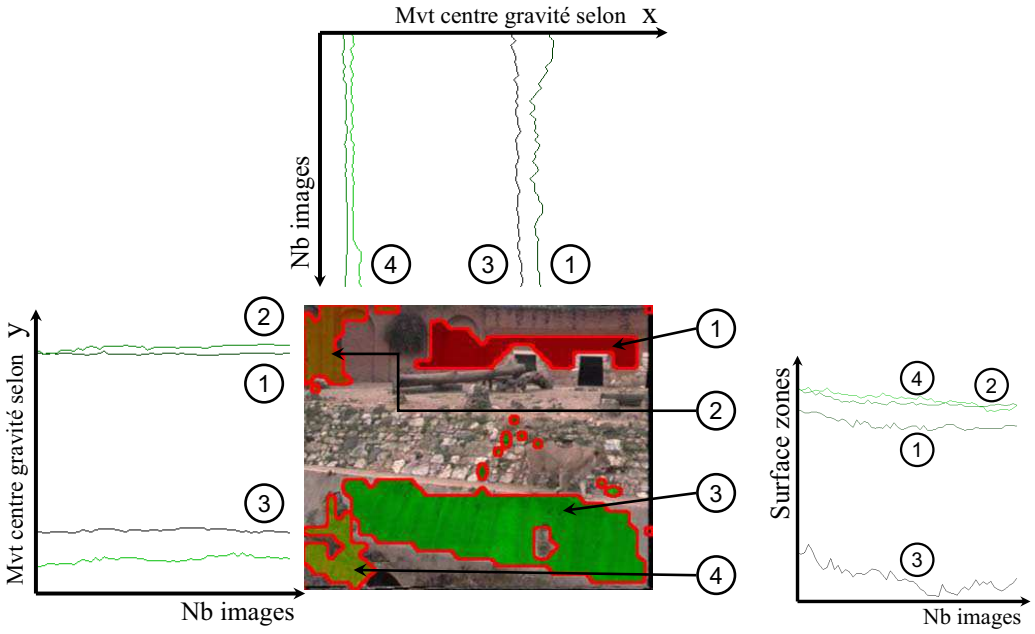


FIG. 2.17 : Mouvement des centres de gravité et variation de la surface des zones de même couleur au cours du temps pour la séquence *Lion*.

2.8 Conclusion

Dans ce chapitre, nous avons posé une distance colorimétrique, tout d'abord sur une seule image mosaïque, que nous avons étendue à une succession d'images

[1] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*

mosaïques en supposant une corrélation temporelle. Les résultats ainsi obtenus le sont en quasi temps réel.

L'utilisation des contours actifs appliqués aux *B-Splines* (utilisation de résultats obtenus au sein de notre équipe) nous permet d'obtenir un affinage de la segmentation au prix d'une augmentation significative du temps de calcul.

Chapitre 3

Conclusion

Après plusieurs itérations, nous parvenons à une segmentation des zones de couleur uniforme. Il faut faire attention lors de chaque itération à ne pas agglomérer des zones de couleur uniforme qui n'ont rien à voir entre elles. En effet, notre méthode ne peut pas revenir en arrière ; lorsque nous obtenons une grande zone, il n'est plus possible de la découper, ce qui explique l'importance du choix des seuils.

Le fait de travailler sur des images mosaïques altère la précision du travail. En effet, utiliser uniquement la moyenne de chaque bloc pour effectuer les estimations induit un certain manque d'information, et nous avons vu que la présence d'une forte texture peut nous poser des problèmes (sur-segmentation du résultat).

De plus, avec l'utilisation des contours actifs appliqués à des *B-Splines*, nous avons obtenu un raffinement du résultat au prix d'une augmentation du coût de calcul. Par contre, il est ainsi possible de désagglomérer des blocs qui avaient été agglomérés par erreur ; toutefois, il ne peut y avoir de création de nouvelles zones. Enfin, sans passer au niveau du pixel, il est possible d'avoir des contours plus proches des zones réelles, car dans la nature le monde est plutôt courbe que droit.

Perspectives

La première perspective est d'utiliser le flux optique dans le critère de la distance colorimétrique 2D+t. En effet, nous avons regardé le 25 voisinage dans les images mosaïques suivante et précédente, mais nous avons, par le flux *MPEG*1-2, pour presque tous les blocs un vecteur mouvement qui décrit son mouvement à travers le temps. Pour chaque bloc, en fonction du poids de l'image d'erreur (plus ou moins de termes dans la *DCT*), nous pouvons préciser si le mouvement a bien été estimé. Dans le cas d'une bonne estimation, nous chercherons, dans l'image mosaïque précédente, uniquement dans le voisinage de neuf blocs autour de la position du bloc donné par le mouvement. De ce fait, le 25-voisinage sera réservé aux blocs n'ayant pas de mouvement transmis ou dont le mouvement transmis a généré une forte image d'erreur.

La deuxième perspective est l'utilisation des *splines* pour la régularisation 2D+t. En effet, pour l'instant nous n'avons régularisé les zones 2D+t que zone 2D par zone

2D. Il sera intéressant de régulariser suivant l'axe temporel.

La troisième perspective est qu'il peut être intéressant, surtout sur les images de type Intra, de regarder de plus près les valeurs autres que *DC* de la *DCT*. En effet, ces valeurs nous apprennent beaucoup de choses sur la régularité de la couleur au sein du bloc. Si seule la valeur *DC* est non nulle, nous pouvons dire que le bloc possède une couleur complètement uniforme ; par contre, si les premiers éléments de la *DCT* sont tous différents de zéro, il est probable que la couleur est tout, sauf uniforme.

Maintenant, il reste à agencer les zones entre elles afin de décider leur appartenance au fond ou aux objets en mouvement, ce que nous traitons dans la partie suivante.

Troisième partie
Segmentation des objets en mouvement

« Rien de ce qui nous entoure ne nous est objet, tout nous est sujet »
Le surréalisme et la peinture, André Breton (1896-1966)

Résumé : Après avoir estimé le mouvement de la caméra, dans la première partie, et avoir segmenté l'image en zones de couleur uniforme (zones 2D+t) dans la partie précédente, nous réunissons toutes ces informations afin d'arriver à notre but : la segmentation des objets en mouvement. Un objet sera dit en mouvement s'il ne suit pas celui de la scène (il est en mouvement relatif par rapport à la scène).

Chapitre 1

Fusion des méthodes précédentes

Dans la figure suivante, nous visualisons ce que nous avons pu obtenir lors des traitements mis en place dans les deux parties précédentes (estimation du mouvement et extraction de zones de couleur uniforme). Ce que nous désirons, c'est extraire l'objet en mouvement. Dans cette séquence, Stefan Edberg, extraite du *COST* 211[1], c'est le joueur de tennis.

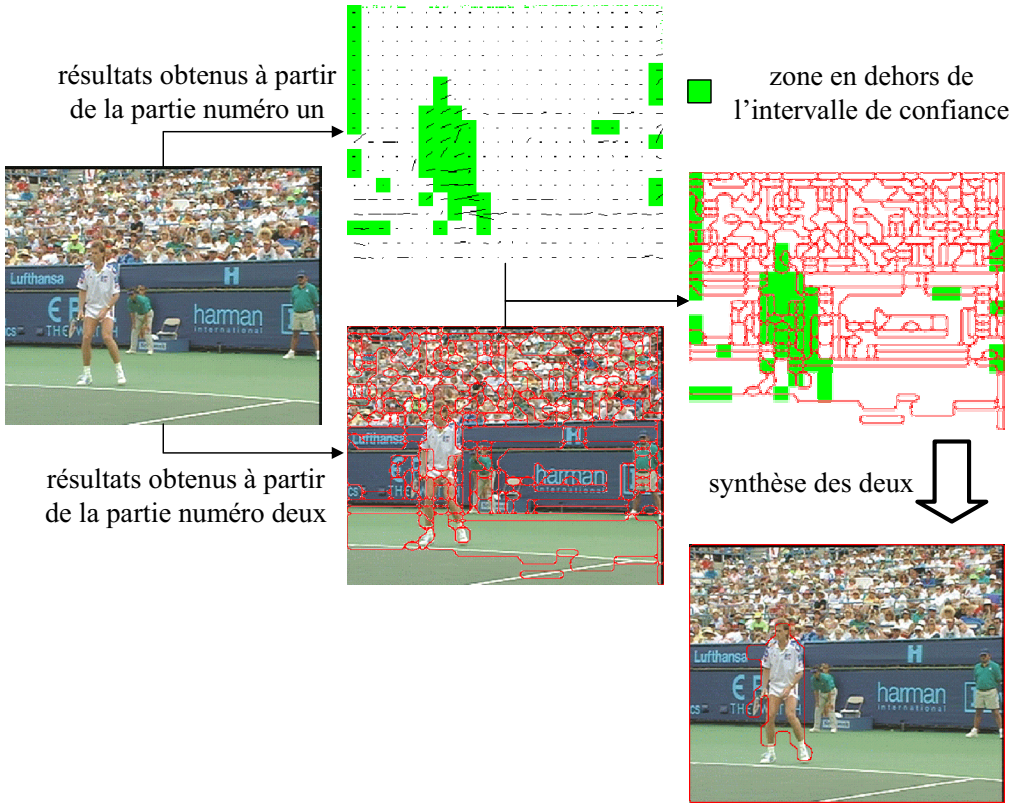


FIG. 1.1 : Sur l'image 51, récapitulatif des données que nous avons obtenues.

[1] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*

Nous avons obtenu :

- de la première partie (à partir de la page 41), nous avons estimé, à partir des vecteurs résiduels (différence entre le vecteur *forward* normalisé et le vecteur idéal qui suit le mouvement estimé de la caméra) les zones en dehors d'un intervalle de confiance et ceci afin d'obtenir le mouvement de la caméra au cours du temps ;
- de la deuxième partie (à partir de la page 87), nous avons défini des zones de couleur uniforme que nous avons extraites et que nous avons pu suivre d'une image sur l'autre.

C'est avec ces informations que nous désirons obtenir la segmentation de l'objet en mouvement (le joueur de tennis).

Mais pour effectuer cette segmentation, fixons-nous tout d'abord le domaine de validité de notre algorithme. Nous partons de séquences animées et non d'images fixes (pour ces dernières, nous vous renvoyons au livre de J.P. Cocquerez et S. Philipp [27] qui donnent un aperçu très détaillé des méthodes qui existent dans ce domaine). Nous utilisons toutes les informations que nous avons analysées dans les deux premières parties et cela de façon simultanées.

À cause de la première partie, notre algorithme de segmentation ne pourra s'appliquer que sur les plans de séquence où l'estimation du mouvement est valide. Dans le cas contraire, d'autres méthodes devront être mises en œuvre, par exemple, nous pouvons citer celles développées au sein de notre équipe et en particulier les récents travaux de S. Jehan-Besson *et al.* [56].

1.1 Algorithme mis en œuvre

Afin de segmenter les objets en mouvement (qui sont la réunions de plusieurs zones de couleur uniforme), nous utilisons les zones 2D+t de couleur uniforme et le mouvement résiduel pour chaque pixel de la zone. Nous attribuons à chaque pixel de chaque zone un poids en fonction de la classification ci-dessous, ce qui nous permet au final de dire si la zone 2D+t, en cours d'étude, fait partie, en totalité, du fond (poids faible), ou bien de l'objet (poids élevé).

Tout d'abord, énumérons les différents types de pixels que nous rencontrons dans une zone :

1. le pixel qui fait partie d'un bloc codé sous forme d'Intra dans une image de type P ou B.
2. le pixel qui fait partie d'un bloc dont le mouvement résiduel est en dehors de l'intervalle de confiance et son bloc d'erreur est 'trop important' ;
3. le pixel qui fait partie d'un bloc dont le mouvement résiduel est en dehors de l'intervalle de confiance et son bloc d'erreur est 'correct' ;
4. le pixel qui fait partie d'un bloc dont le mouvement résiduel est dans l'intervalle de confiance et et son bloc d'erreur est 'trop important' ;
5. le pixel qui fait partie d'un bloc dont le mouvement résiduel est dans l'intervalle de confiance et et son bloc d'erreur est 'correct'.

Nous avons les cinq types de pixels différents avec leur poids en ordre décroissant. En effet, les types 1, 2 et 3 sont des types qui participent à accroître le poids de la zone, en vue de l'étiqueter 'objet en mouvement'. Dans un premier temps, il est possible de fixer un comme pondération aux type 1, 2 et 3 et une pondération nulle pour les types 4 et 5.

À partir de là, avec tous les pixels d'une zone (qui peut s'étendre sur plusieurs images), nous estimons son poids total puis son poids moyen par pixel. En fonction de ce poids, nous attribuons à la totalité de la zone 2D+t le label 'fait partie du fond' ou bien 'fait partie de l'objet'.

Voici l'algorithme de classification des zones de couleur uniforme en pseudo-code :

Algorithme 3 Classification d'une zone 2D+t de couleur uniforme

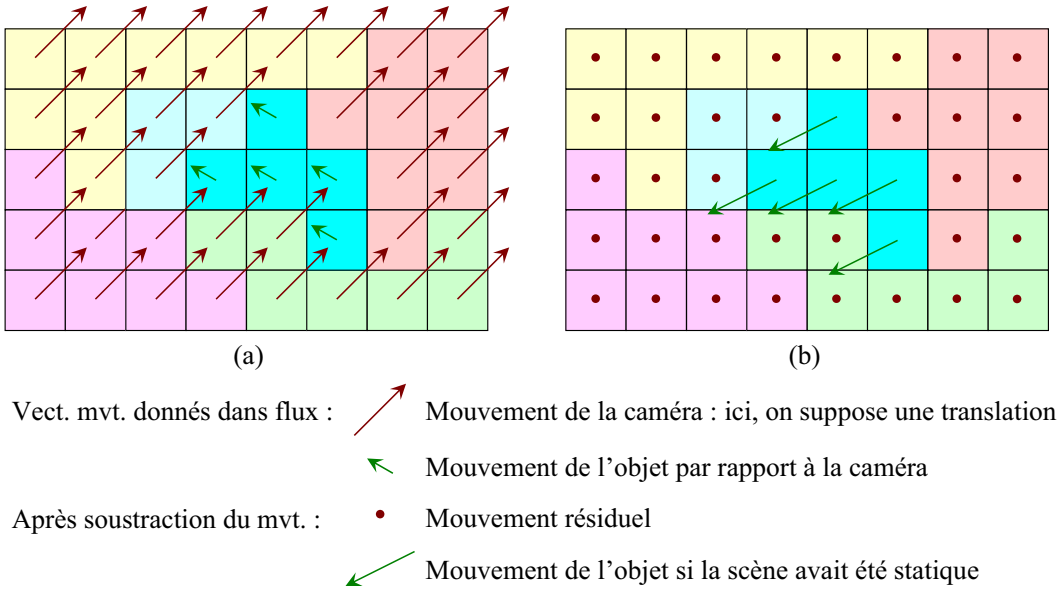
```
soit P1 le poids d'un pixel de type 1
soit ...
soit P5 le poids d'un pixel de type 5
initialiser poidsTotal à zéro
initialiser nbrePixelsDsZone à zéro
pour tout les pixels de la zone '2D+t' faire
|   regarder le type du pixel (de 1 à 5)
|   augmenter poidsTotal du poids du pixel considéré
|   augmenter nbrePixelsDsZone de un
finpour
affecter à poidsMoyen le poidsTotal divisé par
nbrePixelsDsZone
si poidsMoyen est inférieur à un seuil
|   dire que la zone 2D+t fait partie du fond
sinon
|   dire que la zone 2D+t fait partie des objets en mouvement
finsi
```

Regardons de plus près l'étude des vecteurs résiduels puis le résultat de la segmentation, tout d'abord dans un cas idéal, puis dans le cas d'une séquence à caméra fixe et enfin sur des séquences du *COST* 211[2].

Nous nous plaçons sur une image avec des zones de couleur uniforme qui sont des réunions de macroblocs entiers (FIG. 1.2 page suivante).

Tous les vecteurs du flux représentent bien le mouvement réel. Les macroblocs du fond suivent exactement le modèle du mouvement de la caméra et les mouvements de l'objet représentent bien ses mouvements par rapport à la caméra (avec toutes les bonnes suppositions : objet éloigné et plat...). Nous supposons, de plus, que chaque macrobloc appartient, dans sa totalité, soit au fond, soit à l'objet qui a un mouvement différent du mouvement global. Que devrions-nous obtenir ?

[2] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*



(a) vecteurs *Forward* normalisés
(b) après soustraction des vecteurs mouvement donnés par le modèle

FIG. 1.2 : Flux $\mathcal{M}$PEG1-2 avec un mouvement idéal (pour simplifier la représentation, nous nous sommes limités à des mouvements de translations dans le plan perpendiculaire à l'axe optique).

Nous obtenons pour le fond un mouvement résiduel nul (symbolisé par un point) et pour l'objet un mouvement non nul (qui représente le mouvement réel de l'objet si la caméra était restée immobile) (FIG. 1.2-b).

Dans ce cas-là, nous recollons les zones de couleur qui ont le même mouvement (*cf.* article de F. Morier *et al.* [74]) ou tout du moins, nous étiquetons comme zones 'objets en mouvement' les zones qui n'ont pas un mouvement résiduel nul.

Malheureusement, dans la réalité nous ne sommes pas exactement dans ce cas là. Le premier problème vient de ce que le mouvement des macroblocs est estimé par le codeur dont l'objectif est de minimiser le poids des données à envoyer (minimiser l'image d'erreur) et non dans l'optique de donner un mouvement juste (nous en avons déjà parlé dans la partie sur l'estimation du mouvement de la caméra).

Le deuxième problème est qu'un objet en mouvement n'a aucune chance de se retrouver sur la rétine avec uniquement des macroblocs entiers. Si cela peut être vrai au début, cela ne l'est plus lors de son déplacement. En effet, cela impliquerait un mouvement d'au minimum seize pixels (la grandeur d'un macrobloc), ce qui est très important.

Troisième problème, s'il est possible de poser une statistique sur les vecteurs de mouvement du fond (vu leur nombre), il n'en est pas de même pour les objets (nombre de macroblocs trop réduit).

Quatrième problème, à cause du déplacement, il faut gérer les occultations.

1.2 Résultats expérimentaux

Dans les cas réels que nous voyons, nous montrons que le mouvement apporte une partie de la réponse mais pas la totalité. De plus, le passage de zones de couleur uniforme 2D aux zones 2D+t permet une amélioration substantielle du résultat.

1.2.1 Sur une séquence à caméra fixe

Lors de l'estimation du mouvement de la caméra, nous avions enlevé, au fur et à mesure de nos itérations, des macroblocs qui avaient un mouvement non conforme au mouvement de la caméra (*cf.* algorithme page 64). Voici ce que nous obtenions (FIG. 1.3) en supposant à chaque itération que les vecteurs résiduels sont de distribution gaussienne, que les macroblocs qui sont gardés ont leur vecteur résiduel qui est dans les 90% pour l'intervalle de confiance sur les deux axes indépendamment ; les itérations s'arrêtant lorsque les blocs rejetés représentent moins de 10% des blocs restants ou que la surface des macroblocs restants représentent moins de 30% de l'image initiale.

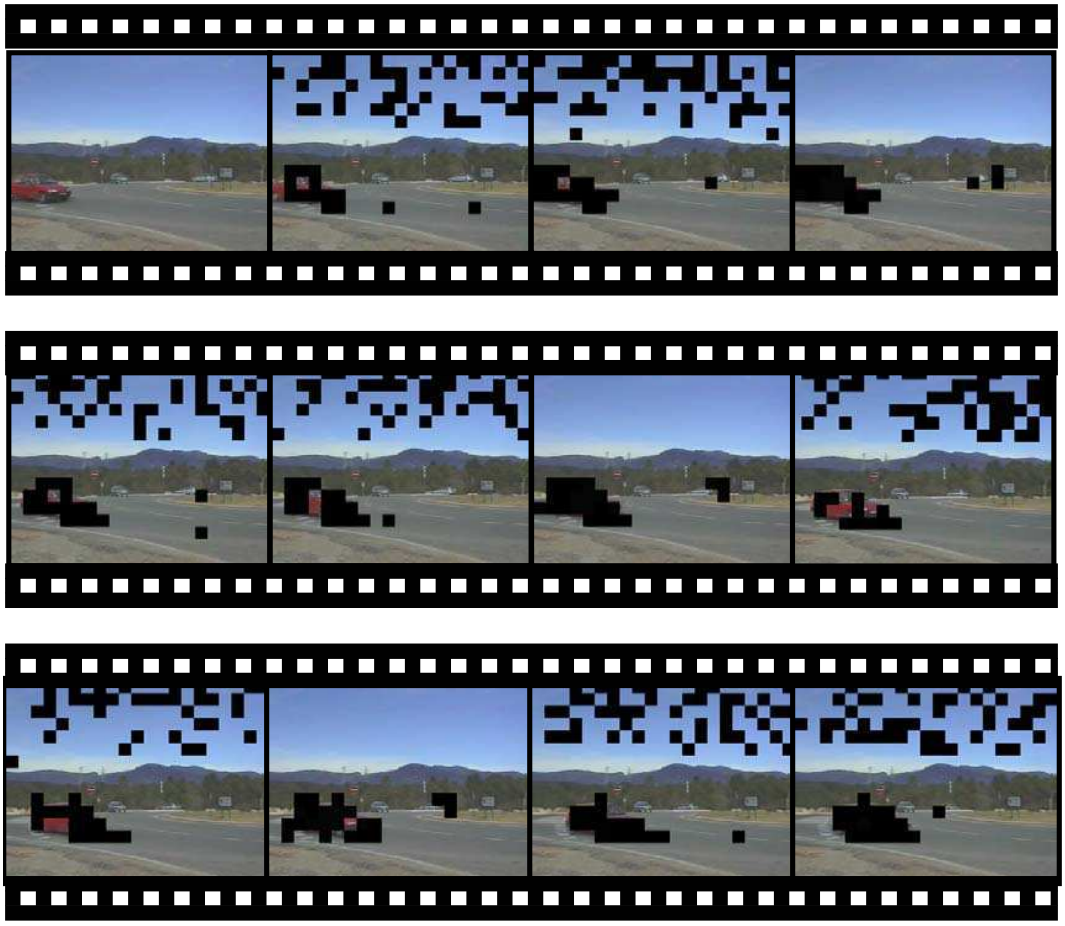


FIG. 1.3 : Zones rejetées lors de l'estimation du mouvement de la caméra.

Que pouvons-nous dire visuellement ?

Tout d'abord, une grande partie des macroblocs de la voiture en mouvement sont bien marqués comme ayant un mouvement non conforme au mouvement de la caméra. De plus, nous remarquons que sur les images de type B, nous avons des macroblocs dans le ciel non conformes au mouvement de la caméra. Ce problème ne se pose pas avec d'autres codeurs. Ce sont donc les aléas du codeur dont nous ne sommes pas maîtres.

Maintenant utilisons les zones de couleur uniforme, pour cela utilisons l'algorithme 3 page 119. Dans un premier temps, nous utilisons uniquement les zones de couleur uniforme 2D, avec cela nous obtenons le résultat de la figure 1.4-a. Si nous passons aux zones 2D+t, nous obtenons le résultat 1.4-b. Le passage du 2D au 2D+t nous permet une amélioration notable du résultat.

Cette amélioration est surtout due aux zones d'occultations qui passent à une étiquette 'fond' dans le cas 2D+t. En effet, dans le cas 2D, ces zones, si elles ne sont pas incluses dans des zones de couleur uniforme plus grandes, risquent d'être étiquetées 'objets' car elles ont un mouvement résiduel qui est éloigné de zéro. Dans le cas 2D+t, le nombre des pixels étiquetés comme 'objets' va devenir minoritaire, et grâce à un seuil bien choisi (par exemple, un poids moyen inférieur à 0,5 si nous avons donné un poids de un pour les pixels de type 1 à 3 et zéro pour les autres), la totalité de la zone va passer dans le camp du 'fond'.

(a) si zone uniquement 2D (b) si passage au 2D+t

FIG. 1.4 : Amélioration lors du passage des zones 2D aux zones 2D+t avec un poids moyen maximal pour le fond fixé à 0,5

Voici le résultat de la segmentation :

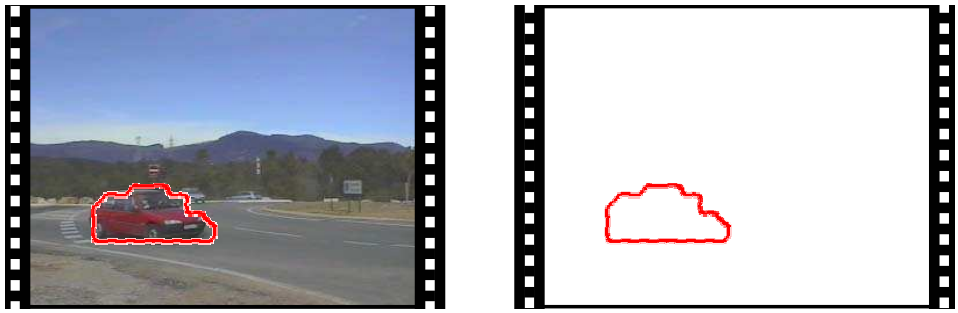


FIG. 1.5 : Résultats de la segmentation sur notre séquence à caméra fixe

1.2.2 Sur la séquence Stefan Edberg du *COST* 211

Tout d'abord, visualisons l'objet en mouvement extrait par notre algorithme avec des zones de couleur 2D+t :

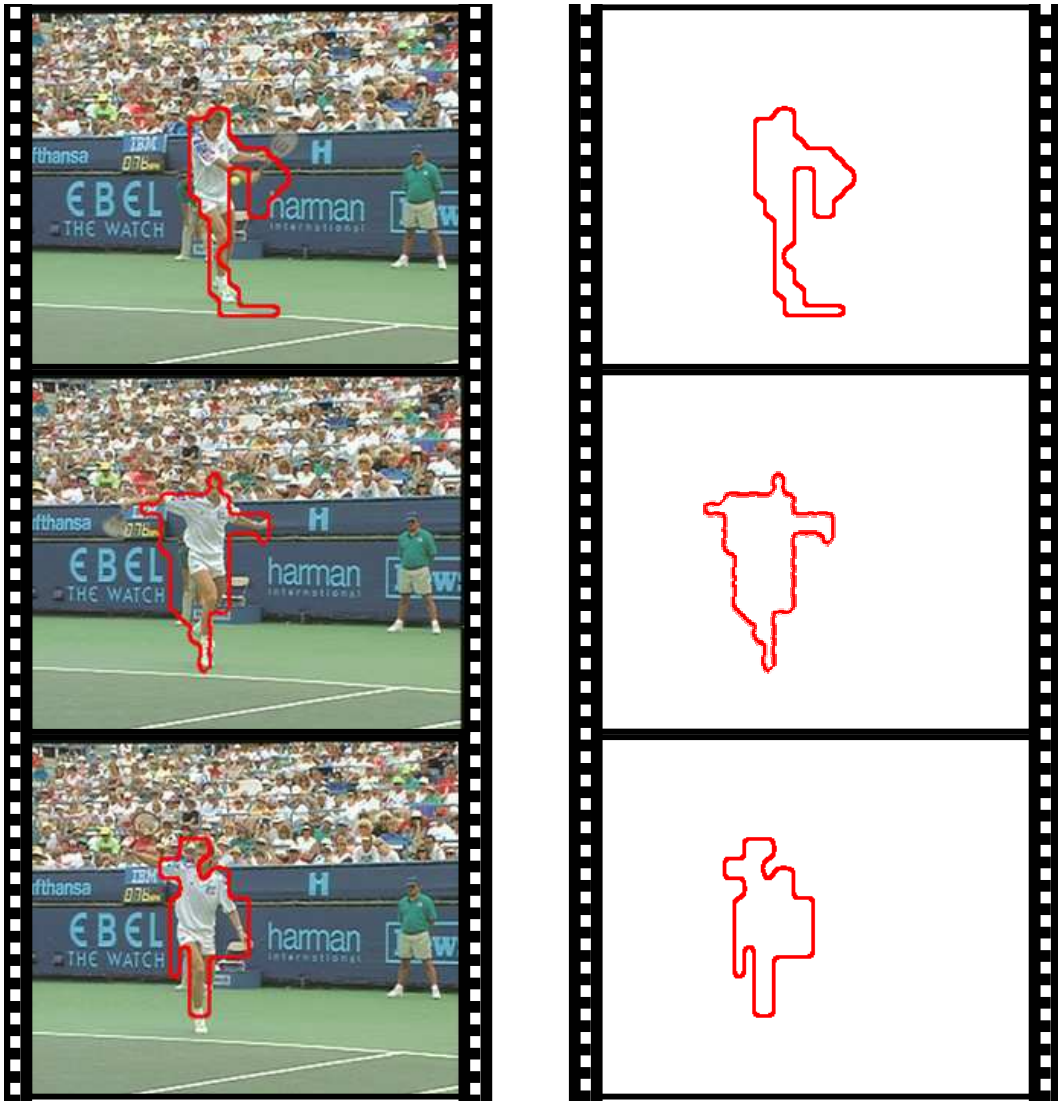
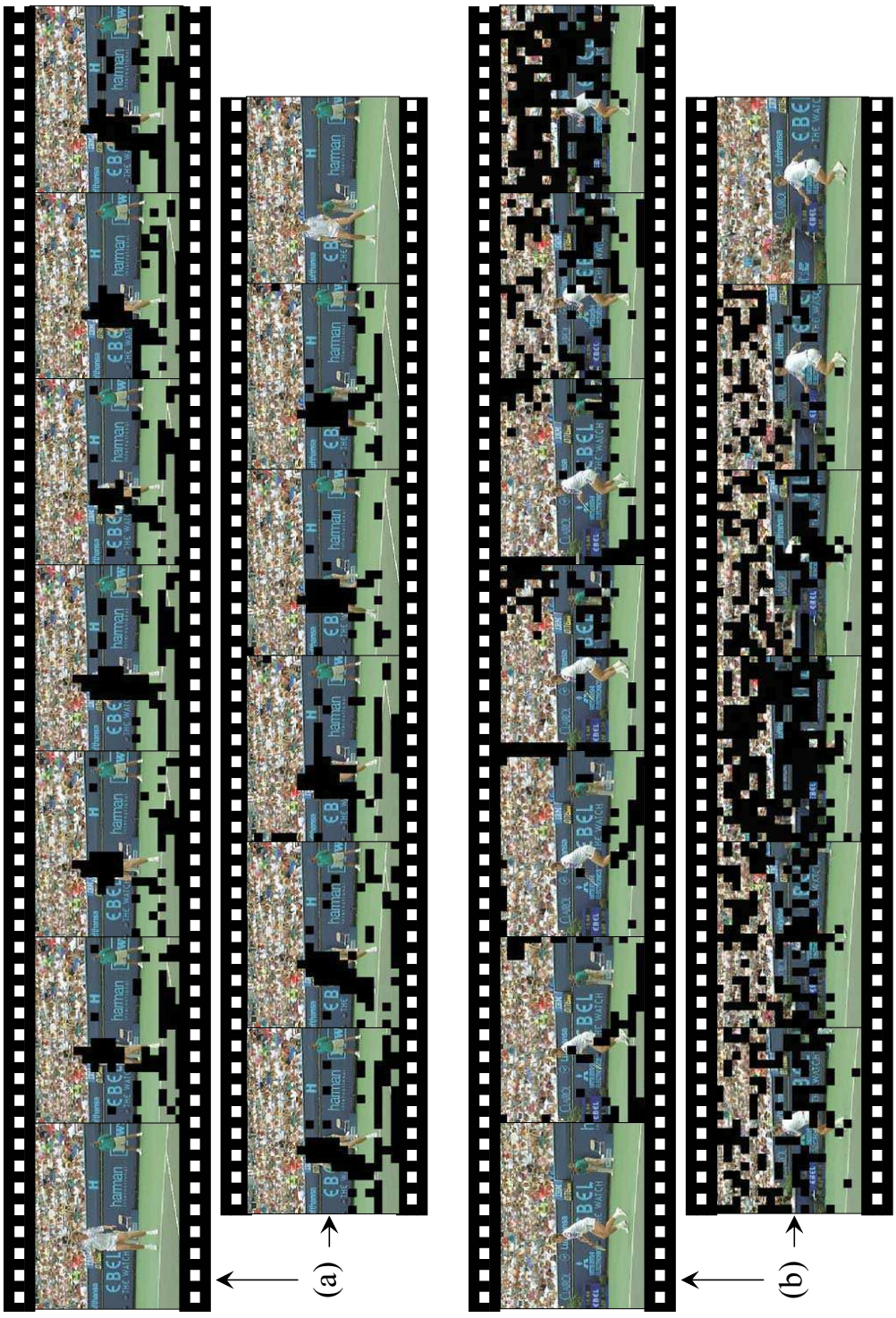


FIG. 1.6 : Résultats de la segmentation sur la séquence Stefan Edberg pour les images 19, 25 et 31.

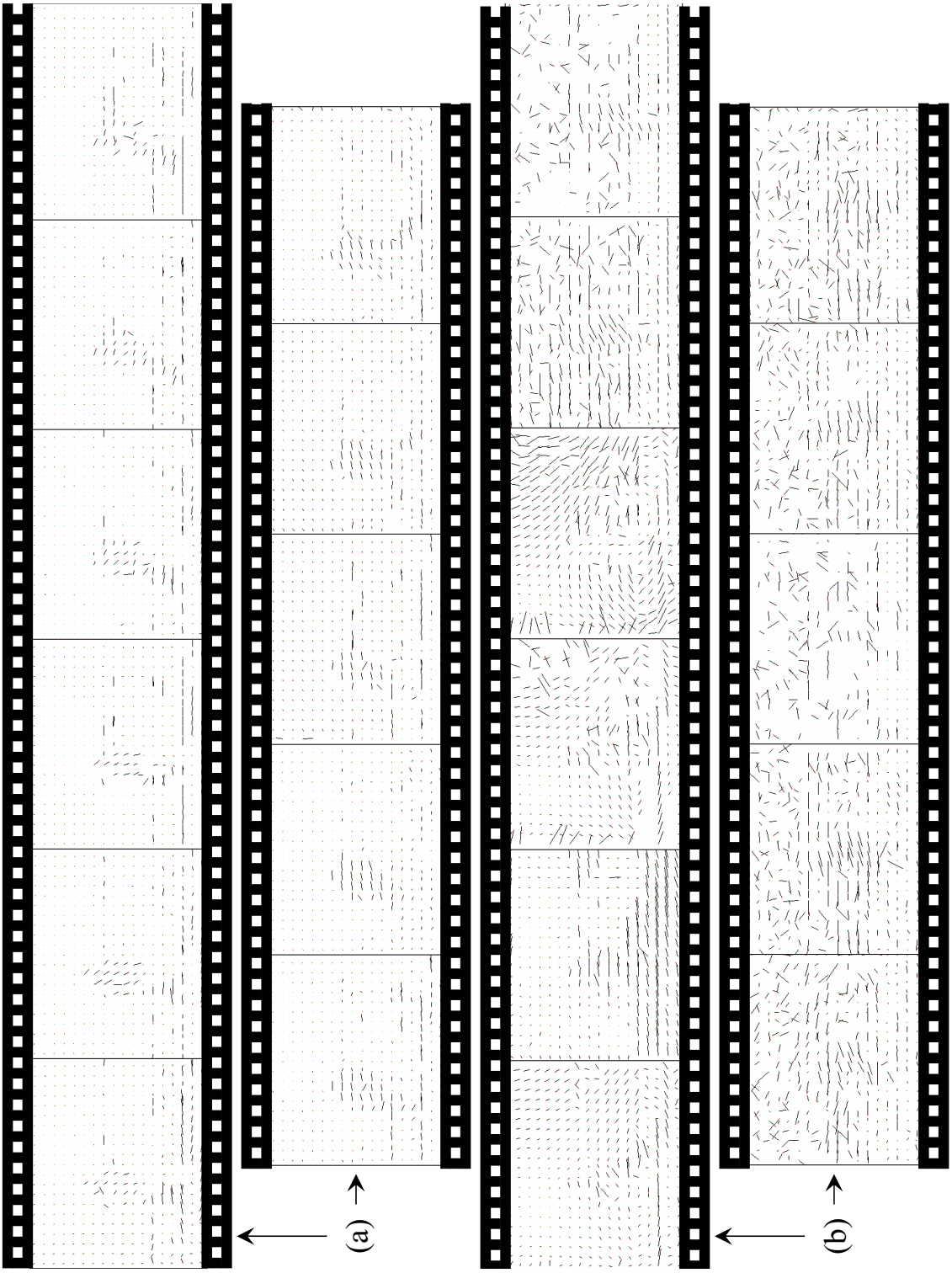
Comme dans l'exemple 'voiture' précédent, nous affichons les macroblocs étiquetés comme non pertinents au cours de l'estimation du mouvement de la caméra (car en dehors de l'intervalle de confiance fixé). Visualisons en noir ces macroblocs dans la figure 1.7 page suivante pour la séquence Stefan Edberg avec un débit de consigne de 1,1 Mbits/s et une recherche revenant à une prédiction de mouvement limitée à sept pixels entre deux images. Pour la partie (a) de la figure, nous avons le *GOP* qui débute à la 37^{eme} image et se termine à la 49^{eme}, pour la (b), c'est entre les images 85 et 97. Nous sommes dans un *GOP* de type *IBBPB....*

Visuellement : maintenant, connaissant les endroits où nous obtenons une bonne prédiction de mouvement, que pouvons-nous dire, visuellement ? La partie (a) est dans une partie où le mouvement a bien été prédit. Nous remarquons que les macroblocs en dehors de l'intervalle de confiance se situent surtout là où se trouve le joueur de tennis mais aussi sur les zones où il y avait occultation et pour terminer sur les zones qui appartiennent au fond mais qui ne sont pas perpendiculaires à l'axe optique (ce que nous avions supposé pour l'estimation du mouvement de la caméra). Le reste de l'image, dans sa plus grande partie, donne des macroblocs qui suivent le mouvement global de la caméra. Pour la partie (b), partie dans laquelle le mouvement de la caméra est mal estimé, nous remarquons, visuellement, que ces macroblocs ne sont pas placés à un endroit privilégié (pas plus sur le joueur qu'ailleurs).



(a) images 37 à 49 - (b) images 85 à 97

FIG. 1.7 : Sur deux *GOP* de la séquence Stefan Edberg, affichage (en noir) des macroblocs dont les vecteurs sont en dehors de l'intervalle de confiance lors de l'estimation du mouvement de la caméra.



(a) images 38 à 48 - (b) images 86 à 96

FIG. 1.8 : Sur deux *GOP* de la séquence Stefan Edberg, affichage des vecteurs résiduels

Lorsque nous avons étiqueté un macrobloc comme non pertinent, cela signifiait que son mouvement résiduel n'était pas faible ; c'est ce que nous confirme l'étude de la figure 1.8 page ci-contre. Dans cette figure, nous avons enlevé les images de type *I*. Que pouvons-nous dire ?

Visuellement : dans le (a), nous avons bien pour le fond un mouvement résiduel quasi nul sauf pour le sol (zone parallèle à l'axe optique) qui a un mouvement non nul mais surtout différent du mouvement du joueur. Dans le (b), le mouvement résiduel est important, cela confirme que le mouvement trouvé n'est pas le bon et qu'il faut mettre en œuvre d'autres techniques.

1.2.3 Résultats sur d'autres séquences du *COST* 211

Séquence Garde-côtes

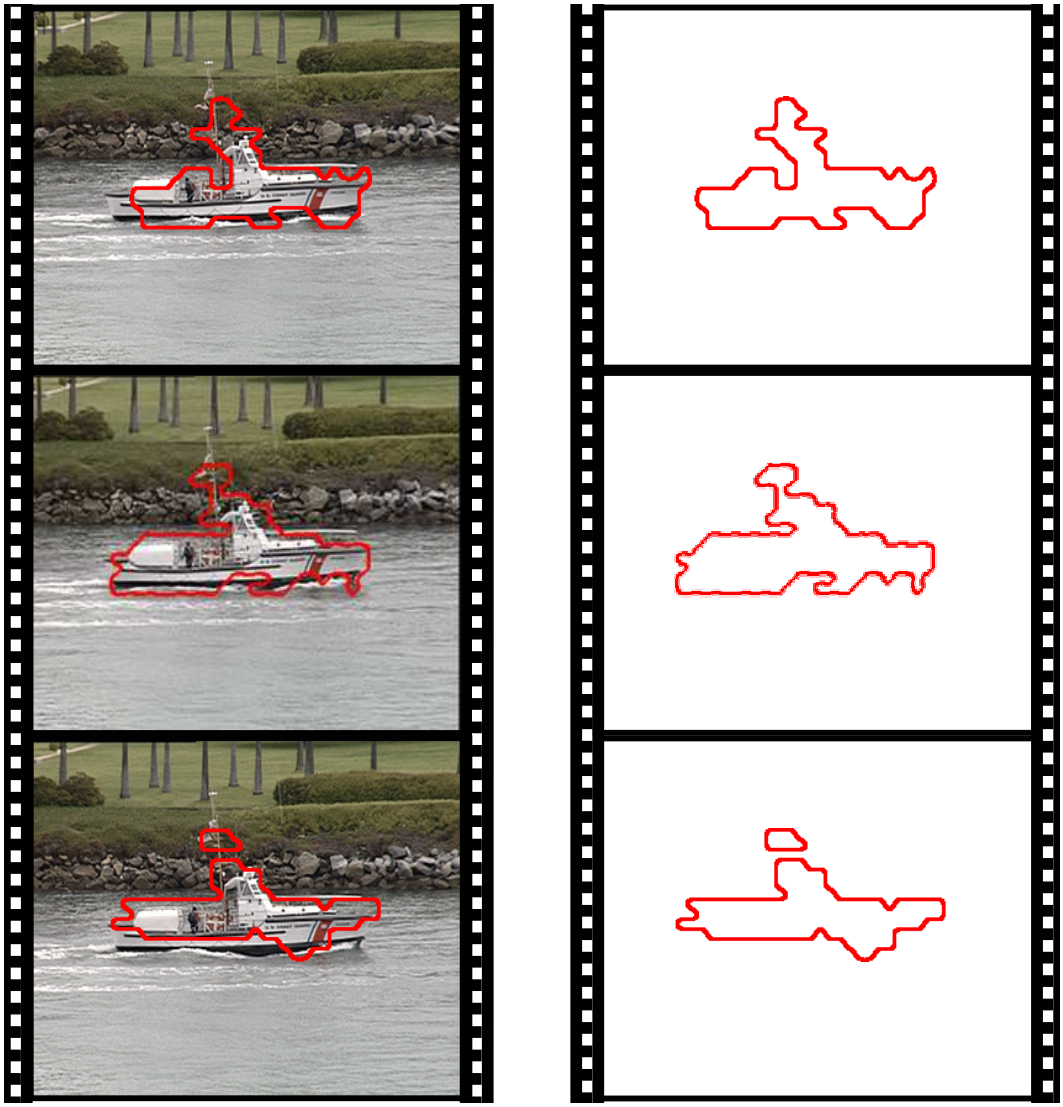


FIG. 1.9 : Résultats de la segmentation sur la séquence Garde-côtes pour les images 256, 262 et 268.

Séquence Hall

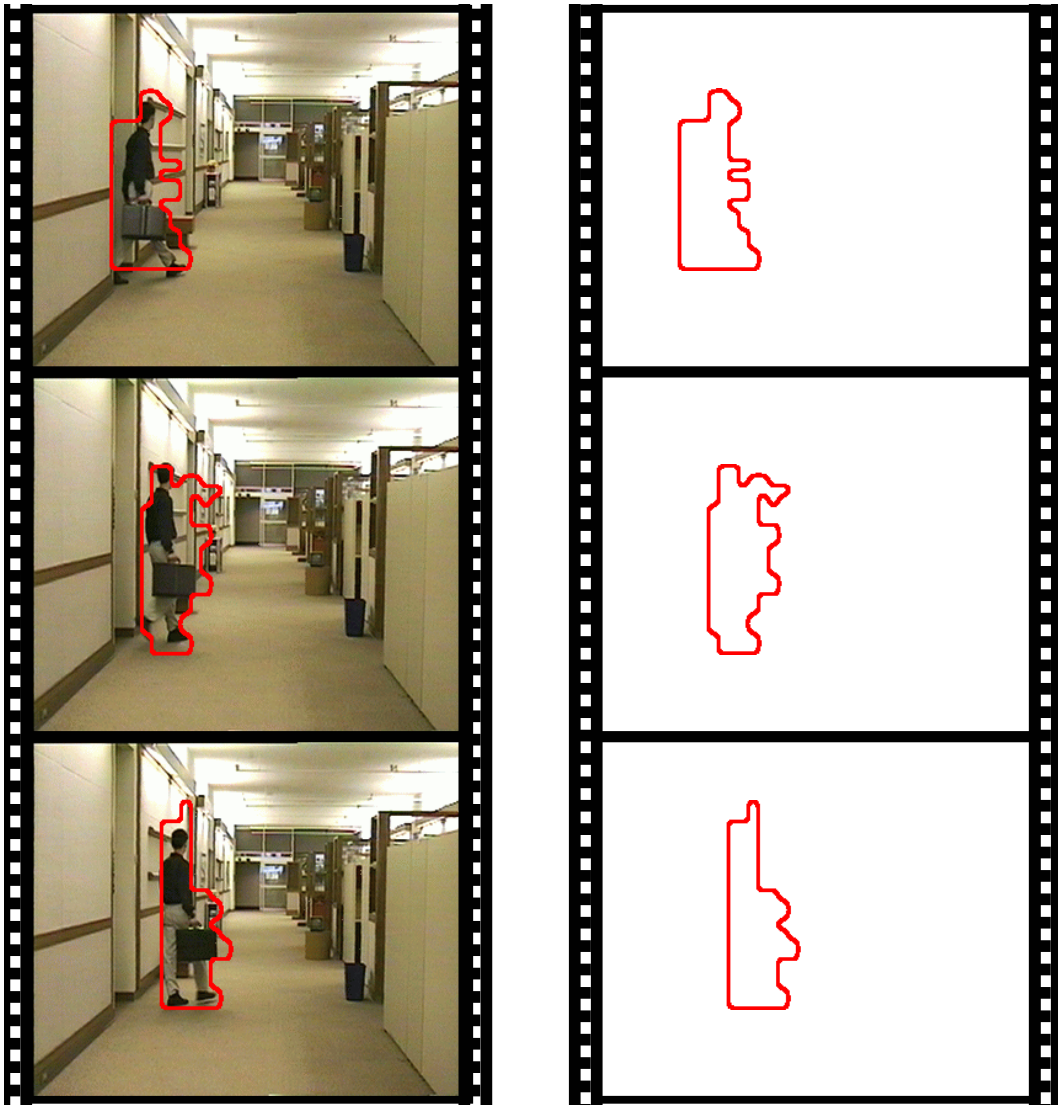


FIG. 1.10 : Résultats de la segmentation sur la séquence Hall pour les images 27, 33, 39.

1.3 Suivi des objets en mouvement

Ce que nous désirons maintenant c'est trouver le mouvement relatif, par rapport au fond, des objets étiquetés comme 'non fond' dans la séquence. Grâce à l'utilisation de l'intervalle de confiance vu lors de l'étude de la première partie, nous avons pu étiqueter les macroblocs comme 'fond' et 'non fond'. Vu que le mouvement des objets en mouvement peut être supérieur au mouvement maximum autorisé et vu que la surface des objets est faible (par hypothèse), il n'est pas possible de leur appliquer une statistique. Le mouvement des objets ne pourra donc pas être estimé à

partir des vecteurs donnés dans le flux. Par contre, nous avons le mouvement des zones de couleur uniforme au cours de la séquence. En prenant les zones (2D+t) qui ont été étiquetées comme ‘non fond’, nous pouvons agréger les zones ayant un mouvement de zone à travers la séquence proche.

À partir de là, afin de remplir certains champs demandés par $\mathcal{M}$PEG7 (*cf.* annexe A page 148), après avoir extrait les objets pour chaque image, il est possible de les entourer chacun par une boîte englobante (FIG. 1.11). Nous appelons boîte englobante un rectangle entourant le contour, et qui minimise sa surface.

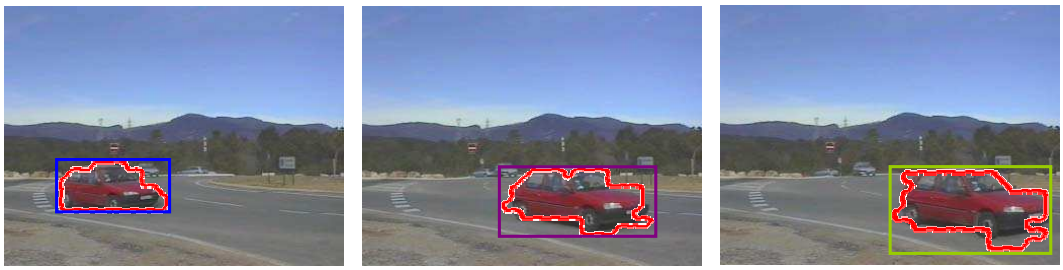
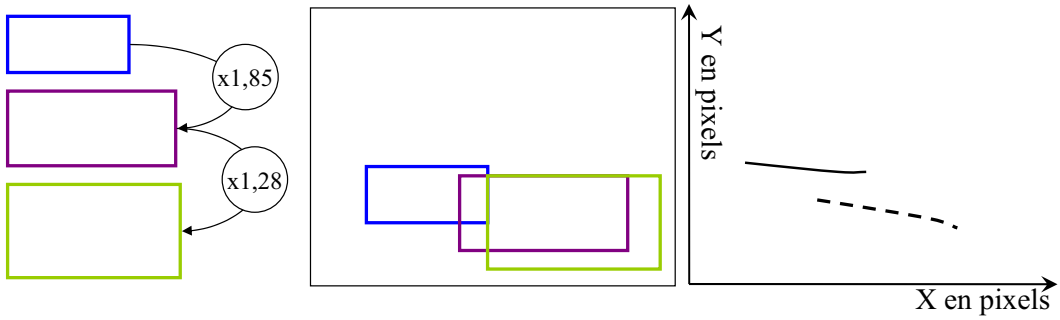


FIG. 1.11 : Boîte englobante sur la séquence voiture.

Avec cette boîte englobante, il est possible de suivre son mouvement à travers la séquence, ce qui nous permet de donner les mouvements de translation pour l’objet. Une étude de la déformation de la boîte nous permet de déduire un rapprochement ou un éloignement de l’objet (FIG. 1.12) ainsi qu’une rotation de l’objet, perpendiculairement à l’axe optique.



(a) coefficient de grossissement ;
(b) positionnement des boîtes englobantes au cours du temps ;
(c) en trait plein : mouvement du coin haut gauche de la boîte englobante - en pointillé : mouvement de son centre de gravité.

FIG. 1.12 : Suivi de l’objet à travers le temps.

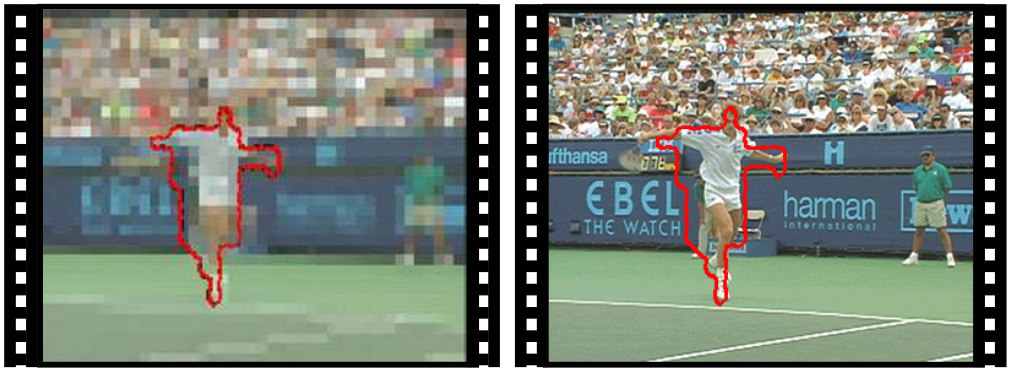
Après le mouvement de la caméra, ces informations de mouvement et de déformation des objets permettent de renseigner de nouveaux champs $\mathcal{M}$PEG7.

Chapitre 2

Conclusion

Dans cette troisième partie, nous avons utilisé toutes les estimations que nous avions faites au préalable pour arriver à notre but : l'extraction des objets en mouvement.

Dans les zones de validité de l'estimation du mouvement de la caméra, le résultat que nous obtenons est visuellement correct. La segmentation n'épouse pas complètement l'objet, ce qui est normal, car nous ne sommes pas au niveau du pixel mais au niveau du bloc. En effet, nous travaillons sur des images mosaïques, et si nous affichons les contours des objets sur l'image réelle, cela sert juste à notre visualisation (FIG. 2.1).



(a) (b)

(a) Image de travail utilisée par le programme ;
(b) Visualisation du contour sur l'image initiale afin d'estimer visuellement la pertinence du résultat.

FIG. 2.1 : Travail sur une image mosaïque (séquence Stefan Edberg image 25).

En ayant pu suivre les zones de couleur uniforme à travers la séquence, il est possible de suivre l'objet en mouvement. Ce suivi permet de déduire son mouvement et son évolution (grossissement, rotation selon l'axe optique...) en vue de renseigner certains champs demandés par $\mathcal{M}$PEG7.

Perspectives

Comme première perspective, il sera intéressant de comparer nos résultats avec ceux des méthodes, qui travaillent au niveau du pixel, développées au sein de notre équipe. Pour cela, nous développerons un critère de qualité qui compte le nombre de pixels bien étiquetés et ceux mal étiquetés.

La deuxième perspective, ayant l'objet en mouvement qui est extrait, il est possible de l'indexer, par exemple par sa signature couleur.

Conclusions et Perspectives

"Life is the art of drawing sufficient conclusions from insufficient premises"[1]
Samuel Butler (1835-1902)

Résumé : si ce n'était que le début...

[1] « La vie est l'art de tirer des conclusions suffisantes de prémices insuffisantes. »

Dans ce manuscrit, nous avons vu des méthodes permettant d'estimer, et ceci en temps réel, le mouvement de la caméra sur une séquence d'images ainsi que les méthodes permettant d'extraire les objets en mouvement, toujours en quasi temps réel. Pour ce faire, nous avons utilisé une séquence vidéo codée selon la norme $\mathcal{M}$PEG1-2. Nous avons extrait du flux le maximum d'informations ayant déjà été traitées lors du codage (en évitant le décodage total du flux). Le fait que les résultats viennent du codeur implique que nous soyons tributaires et ignorants de la pertinence du traitement.

Dans la première partie, nous avons étudié l'estimation du déplacement de la caméra, pour l'instant limitée à quatre mouvements (les translations horizontale et verticale dans le plan de la rétine, le zoom et la rotation axiale). En utilisant un modèle affine simplifié à quatre paramètres, les résultats obtenus, et très rapidement, sont assez proches de la réalité ($\pm 0,5$ pixel) si le mouvement global est inférieur au mouvement maximal utilisé lors du codage. Afin d'évaluer la pertinence du résultat, nous avons étudié les images d'erreur. En effet, si elles sont lourdes ou s'il y a trop de macroblocs codés sous forme d'Intra, nous constatons une mauvaise prédiction du mouvement et donc, une mauvaise estimation du mouvement de la caméra. Dans ce cas là, il faut mettre en œuvre d'autres méthodes plus longues, nécessitant la décompression complète du flux mais qui donnent, à coup sûr, un résultat plus proche de la réalité. À partir de là, nous pouvons commencer à renseigner certains champs de $\mathcal{M}$PEG7 concernant le mouvement de la caméra au cours de la séquence (pour l'instant, uniquement quatre mouvements sur les sept attendus).

Dans la deuxième partie, nous avons traité de la couleur. Nous avons montré différents espaces de couleur, tout d'abord l'espace RVB qui est performant pour la représentation d'images sur un écran mais aussi l'espace L-Cr-Cb qui est mieux adapté à la vision humaine (cet espace est utilisé dans la trame $\mathcal{M}$PEG1-2). Avec ces informations colorimétriques, nous avons segmenté l'image en zones de couleur uniforme. Pour ce faire, nous avons commencé avec l'introduction d'une distance colorimétrique. Deux blocs, qui peuvent être dans deux images consécutives (aspect 2D+t), seront dans la même zone si leur distance est inférieure à un seuil. Puis, afin d'améliorer les résultats et comme le contour d'une zone dans une image est plutôt courbe que droit, nous avons utilisé les *B-Splines* dont le contour est défini pour s'adapter à la forme de l'objet afin d'obtenir des zones de couleur uniforme les plus proches de la réalité (ce qu'un opérateur humain donnerait). Cette dernière méthode permettant d'affiner le résultat au prix d'une augmentation importante du temps de calcul (nous ne sommes plus dans le temps réel).

Enfin, dans la troisième partie, nous avons réuni toutes les informations qui ont été données dans les parties précédentes pour obtenir la segmentation des objets en mouvement. Un objet sera dit en mouvement s'il ne suit pas celui de la scène (il est en mouvement relatif par rapport à la scène). Le fait de pouvoir le suivre à travers la séquence, va nous permettre de renseigner d'autres champs $\mathcal{M}$PEG7.

À part l'utilisation des *B-Splines* qui nous fait quitter le temps réel, toutes nos méthodes opèrent rapidement et permettent de renseigner certains champs demandés par $\mathcal{M}$PEG7. Il est possible, maintenant, d'une partie de séquence où le mou-

vement est continu, d'extraire une image clef en vue de l'indexer.

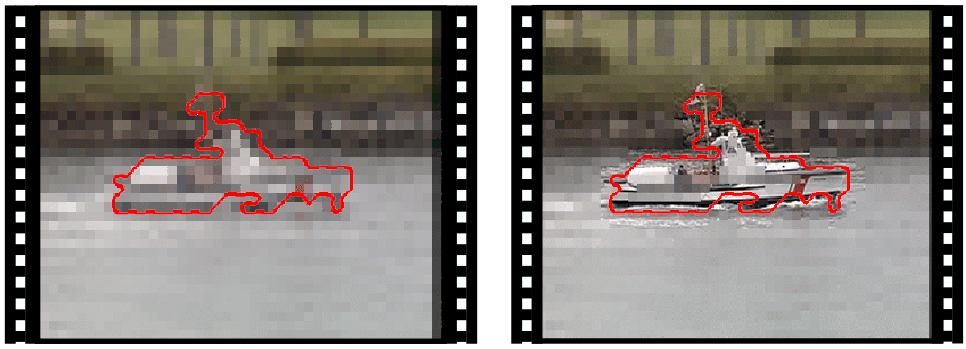
Perspectives

Nous pouvons dégager quelques perspectives au travail présenté ici.

Tout d'abord, créer un critère de qualité, qui permettra de déterminer les pourcentages de pixels bien et mal étiquetés.

Dans un deuxième temps, il faudra améliorer l'utilisation de *B-splines* afin d'utiliser le plan temporel pour régulariser les formes de chaque zone sur les images.

Puis, toujours pour améliorer le résultat de la segmentation, et toujours dans l'optique d'une segmentation en temps réel, il pourrait être intéressant de décompresser une bande autour de l'objet que nous avons détecté. De ce fait, il ne serait pas nécessaire de décompresser tout le flux, mais avec le contour initial que nous avons trouvé, nous pourrions obtenir un contour encore plus fin en le faisant varier (FIG. 2.2).



(a) résultat de la segmentation des objets en mouvement

(b) décompression d'une bande autour du contour

FIG. 2.2 : Bande autour de l'objet segmenté.

Maintenant que nous connaissons les objets en mouvement, il faudra les indexer dans une base de données. D'un côté nous avons notre objet et de l'autre une base qui a été indexée. Il faut mettre en correspondance notre objet avec un objet déjà connu. Seulement il faut décrire le contenu de l'image représentant l'objet. Pour ce faire, nous pouvons utiliser divers attributs, tels que la couleur, la texture ou la forme. Chacun de ces attributs peut avoir diverses représentations comme des histogrammes pour la couleur, par exemple. Ces représentations sont traduites numériquement par des signatures qui sont généralement des vecteurs de dimension très élevée. De nombreux attributs généraux ont été utilisés pour cette tâche, comme dans l'article de M. de Marsicoi *et al.* [68]. Citons notamment la couleur, la texture, la forme qui sont des attributs visuels de bas niveau, mais aussi les images propres, comme dans l'article de S. Sclaroff *et al.* [86], ou l'*ISS* (*Image Shape Spectrum*, soit l'éventail des images de formes) vu dans l'article de C. Nastar et M. Mitschke [77] qui sont des attributs de haut niveau. D'autres techniques combinant des attributs visuels et textuels de l'image ont été développées comme dans l'article de M.J. Swain

et al. [106]. Pour accroître la pertinence du résultat, il sera souvent nécessaire de recoder l'image ou une partie de l'image que l'on désire indexer. En effet, l'espace de couleur L-Cr-Cb n'est pas le mieux adapté. Il est possible, lors de ce recodage, d'utiliser le codage fractal afin d'optimiser l'indexation (articles de A.E. Jacquin ou M. Vissac *et al.* [51, 113, 112]).

Toujours dans l'optique de l'indexation, il peut être intéressant de reconnaître les personnes présentes sur la séquence. Comme nous les avons segmentées, il est possible en décompressant juste cette partie de l'image de mettre en œuvre des techniques de reconnaissances de visages, comme dans l'article de F. Perronnin *et al.* [87].

Annexes

« *Nulla placere diu nec vivere carmina possunt,*
quae scribuntur aquae potoribus. »[2]
Épîtres, I, XIX, 2-3, Quintus Horatius Flaccus (65 - 8 av. J.-C.)

Résumé : Dans la première annexe, nous abordons les normes $\mathcal{M}$PEG. Tout d'abord, d'où vient le nom $\mathcal{M}$PEG ? Et pourquoi la nécessité d'avoir une norme de compression vidéo ? Nous étudions de plus près la partie vidéo des normes $\mathcal{M}$PEG1-2 et nous montrons la structure de la trame avec sa syntaxe. Dans un deuxième temps, nous effectuons un rapide tour d'horizon des possibilités qu'offre la norme $\mathcal{M}$PEG7, surtout sur les points qui nous intéressent.

Dans l'annexe suivante, nous voyons d'autres espaces de couleur comme ceux plus proches de la vision humaine.

Dans la troisième annexe, nous voyons des éléments de calcul pour l'estimation des sept mouvements de la caméra.

Dans la quatrième annexe, nous voyons des courbes complémentaires des estimations du mouvement vues dans la première partie du manuscrit.

Dans l'avant dernière annexe, nous voyons une introduction aux *splines*.

Dans la dernière annexe, nous expliquons un travail que nous avons effectué au sein de l'équipe et qui ne fait pas à proprement parlé partie du sujet de la thèse.

[2]Des vers ne peuvent plaire ni durer longtemps, s'ils ont été écrits par des buveurs d'eau.

Annexe A

Compléments sur quelques normes $\mathcal{M}$PEG

A.1 Pourquoi le nom de $\mathcal{M}$PEG

Le nom $\mathcal{M}$PEG représente l'acronyme de *Moving Picture Experts Group*, soit groupe d'experts pour les séquences animées. Il se nomme pour l'Organisation Internationale de Standardisation et la Commission Internationale d'Électrotechnique : JTC1/SC29/WG11. Ce groupe a établi plusieurs normes. Il y a les normes pour la compression de séquences vidéos ($\mathcal{M}$PEG 1 à 4), une norme pour l'indexation de documents multimédias ($\mathcal{M}$PEG7) et une autre norme, $\mathcal{M}$PEG21 (toujours pour le multimédias). Dans un premier temps, regardons, par un bref historique, les raisons qui ont poussé à la création d'une norme de compression de séquences vidéos. Dans un deuxième temps regardons la norme $\mathcal{M}$PEG1-2 partie vidéo et enfin terminons par un rapide tour d'horizon de la norme $\mathcal{M}$PEG7.

A.2 $\mathcal{M}$PEG pour la compression

Dans le milieu des années 80, plusieurs industriels mettaient au point des technologies pour développer la vidéo numérique. De plus, on voulait aller vers une interactivité plus grande et vers un contenu plus diversifié. Le fait d'avoir une technologie commune permet d'avoir une certaine pérennité. En effet, pour des investissements aussi importants qui allaient être mis en œuvre, il ne fallait pas que le standard, en un endroit, soit différent ailleurs. L'incertitude de la normalisation augmente les coûts et, de ce fait, les risques ; il ne peut résister que les plus grands et pas obligatoirement les plus performants. De plus, c'était le début du câble. Par exemple, en France, c'est en novembre 1982 qu'est lancé le plan câble. Mais comment faire passer le maximum d'informations dans un fil qui devra être soit de la fibre optique, soit du câble coaxial ?

La compression des données devient vitale pour la pérennisation du câble et plus tard du satellite. En effet, le débit n'est pas extensible à l'infini, mais par contre, on

espère accroître le contenu audiovisuel à l'infini...

Il fallait trouver une méthode de compression qui soit normative, facile à décoder. En effet, peut importe que pour coder un film il faille de très gros calculateurs afin de coder en temps réel ; ce qui importe, c'est que, chez le client, le décodage soit facile, rapide et peu coûteux (du point de vue financier).

A.2.1 $\mathcal{M}$PEG1

Étude lancée en 1988 et finalisée en 1991 dans la norme *IS* 11172 (le *IS* pour *International Standard*) qui porte comme intitulé « Technologies de l'information – Codage de l'image animée et du son associé pour les supports de stockage numérique jusqu'à environ 1,5 Mbit/s ».

Cette norme est découpée en plusieurs parties :
- *IS* 11172-1 : c'est la partie qui traite des systèmes ;
- *IS* 11172-2 : c'est la partie qui traite du codage de la vidéo ;
- *IS* 11172-3 : c'est la partie qui traite du codage de l'audio ;
- *IS* 11172-4 : c'est la partie qui traite des essais de conformité ;
- *IS* 11172-5 : c'est la partie qui traite de la simulation de logiciel.

$\mathcal{M}$PEG1 n'a pas complètement répondu aux attentes des professionnels. Pour résoudre cela, le groupe $\mathcal{M}$PEG lance l'étude d'une nouvelle norme qui va devenir la norme $\mathcal{M}$PEG2.

A.2.2 $\mathcal{M}$PEG2

Étude lancée en 1991, syntaxe arrêtée en avril 1993, elle devient une norme internationale en 1995 sous le numéro *IS* 13818 et sous le nom de « Technologies de l'information – Codage générique des images animées et du son associé ». Cette norme a comme ambition une meilleure qualité et une meilleure utilisation de la bande passante. De plus, on rajoute une flexibilité du débit, une possibilité de multi-résolution, un codage des images entrelacées. Ce codage est beaucoup plus mature et puissant (à débit identique, le résultat visuel est meilleur). Il est à noter une comptabilité ascendante des normes ; en effet, un flux $\mathcal{M}$PEG1 est un flux $\mathcal{M}$PEG2 particulier (l'inverse est faux).

Comme dans le cas précédent, cette norme est découpée en plusieurs tomes (plus nombreux que pour $\mathcal{M}$PEG1), les tomes de un à quatre portent sur les mêmes sujets que pour $\mathcal{M}$PEG1.

A.2.3 $\mathcal{M}$PEG3

$\mathcal{M}$PEG3 a bien existé (fugacement), il était destiné à la HDTV (TéléVision Haute Définition) en particulier et puis ce groupe a été fondu dans le groupe $\mathcal{M}$PEG2 (il ne faut pas confondre $\mathcal{M}$PEG3 et MP3 qui, lui, est l'abréviation de $\mathcal{M}$PEG1 audio *layer 3*, soit le troisième niveau de complexité de la compression audio de

$\mathcal{M}$PEG1).

Regardons maintenant l'architecture des normes $\mathcal{M}$PEG1-2 en ce qui concerne la partie vidéo.

A.2.4 Redondances spatiales puis statistiques

Pour chaque bloc, aussi bien pour un bloc Intra que pour un bloc d'erreur (*cf.* page 77), la corrélation entre les valeurs des pixels est forte, d'où l'utilisation d'une transformation espace-fréquence ayant pour but de décorréler les échantillons, et de concentrer l'énergie sur quelques coefficients de la transformée. En effet, dans le domaine fréquentiel, le signal est décrit non plus par des grandeurs physiques les unes à la suite des autres, mais par son spectre, c'est-à-dire l'ensemble des valeurs d'énergie de chacune de ses composantes fréquentielles. Pour cela, la méthode utilisée est une transformée mathématique linéaire et bidirectionnelle que l'on appelle la Transformée en Cosinus Discrète (*DCT* pour *Discrete Cosinus Transform*). La *DCT*, appliquée sur des blocs de taille 8x8 pixels, a pour rôle de convertir les valeurs de luminance ou de chrominance des pixels en coefficients représentant les amplitudes aux différentes fréquences. Or, dans le domaine fréquentiel, la quasi-totalité des informations de l'image se situe dans les fréquences basses du spectre (une image contient généralement une faible proportion de détails très fins), cela veut dire que les composantes énergétiques sont statistiquement plus élevées pour les basses fréquences que pour les hautes fréquences. La *DCT* permet, au final, de regrouper les informations essentielles et de les séparer des données moins significatives. Ce bloc *DCT* de 64 éléments va être réarrangé sous forme d'un vecteur avec au début les basses fréquences (porteuses de beaucoup d'information) et à la fin les hautes fréquences (porteuses de peu d'information).

Après quoi une quantification des coefficients est nécessaire afin de limiter le nombre de bits nécessaires pour les représenter. Cette limitation permet de faire apparaître des zéros au niveau des hautes fréquences.

Le vecteur contient des suites plus ou moins longues de coefficients de valeurs identiques, souvent nulles. C'est pour cela que le codage par plage (*Run Length Encoding* ou *RLE*) est utilisé. Il en résulte une suite d'évènements à répartition non uniforme ; des codes entropiques, traduits par des tables de Huffman pré-calculées par analyse statistique sur des ensembles de séquences tests, sont utilisés pour tirer parti des valeurs les plus fréquentes.

Les redondances spatiales puis statistiques sont évitées par l'utilisation de la *DCT*, suivie d'une quantification et d'un codage entropique.

Abordons maintenant le codage proprement dit.

A.2.5 Syntaxe du flux $\mathcal{M}$PEG1-2

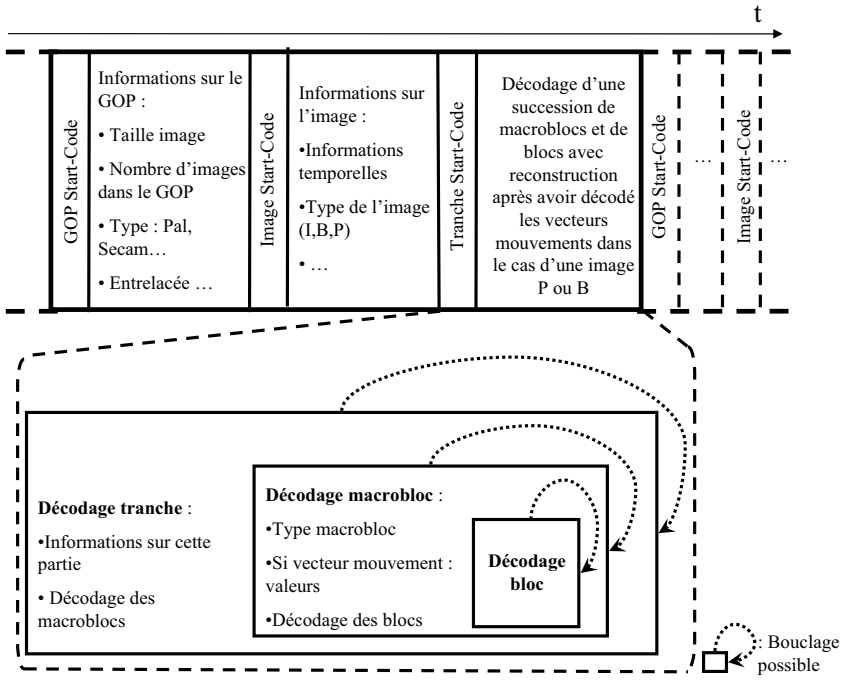


FIG. A.1 : Syntaxe d'un *GOP* du flux $\mathcal{M}$PEG1-2

Le flux se décompose en *GOP* qui sont composés de plusieurs entités (FIG. A.1) :
– les *Start-Codes*
– les en-têtes
– les données proprement dites.

Cette dernière catégorie va des informations sur le mouvement (pour les images de type *P* ou *B*), aux informations sur l'image proprement dite.

Start-Code (code de début)

Les *Start-Codes* sont des codes uniques, codés sur 8 bits. Ce sont des mots réservés qui ne peuvent pas se retrouver dans les mots donnés par le codage entropique. Ils servent à marquer le début et la fin de chaque partie dans le flux. En cas de perte d'information ou d'accès aléatoire dans le flux, ils permettent la re-synchronisation du son et de l'image (FIG. A.2 page suivante). Le pire des cas est la synchronisation au niveau du *GOP* suivant (soit au maximum une demi-seconde de perdue dans le cas d'un *GOP* de 12 images).

Par exemple, nous avons le *Start-Code* de l'image, le *Start-Code* de l'en-tête de la séquence, le *Start-Code* de la fin de la séquence, le *Start-Code* du *GOP*...

Les en-têtes

Concernant les différentes parties du flux :

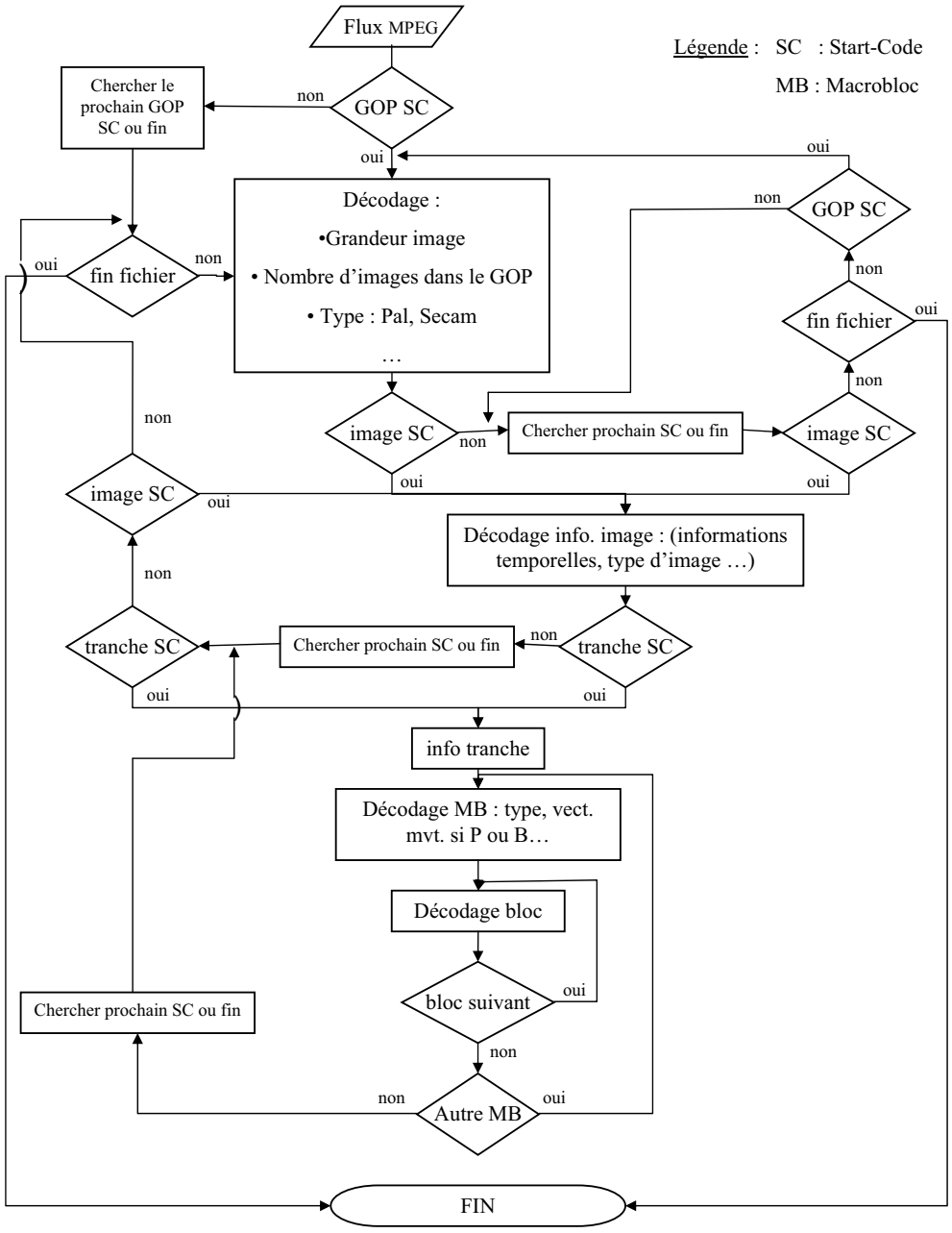


FIG. A.2 : Organigramme succinct du décodeur MPEG1-2 (partie vidéo)

- Pour le *GOP* : l'en-tête contient les dimensions de l'image, l'ordonnancement de celle-ci, le type de la séquence (*PAL*, *NTSC*, entrelacée...), le débit utilisé lors de la compression, des paramètres temporels qui permettent de synchroniser image et son.
- Pour chaque image : l'en-tête contient son type, des paramètres temporels permettant de synchroniser image et son (comme pour le *GOP*). De plus, si c'est un *P* ou un *B,* des informations sur les vecteurs de mouvement.
- Pour chaque tranche : l'en-tête contient quelques informations
- Pour chaque Macroblocs : l'en-tête contient le type du macrobloc (*I*, *P*, *B*)

Ces informations sont nécessaires lors du décodage et leur perte peut rendre inutilisable tout le *GOP* concerné ou tout au moins un bloc ou un macrobloc.

Les données proprement dites

Dans les données proprement dites, nous classons les *VLC* (Codage à Longueur Variable) des *DCT* (dans le cas des blocs) et les *VLC* des vecteurs de mouvement (dans le cas des macroblocs). L'altération, *a fortiori*, la perte d'un *VLC* entraîne sa non décodabilité. Il existe des algorithmes qui permettent la reconstruction de l'information perdue (ils ne sont pas normalisés dans les normes $\mathcal{M}$PEG1-2). Ces algorithmes sont des sujets de recherche.

A.2.6 $\mathcal{M}$PEG1-2 et son utilisation dans le futur

Nous avons vu que cette norme est utilisée pour le *DVD* (*Digital Video Disk*) ou pour la transmission de la télévision *via* le satellite ou le câble. Mais sera-t-elle encore valable avec l'avènement de la télévision via Internet ou via le téléphone portable (*UMTS*) ? En effet, il est nécessaire de pouvoir reconstruire les paquets qui sont perdus ou altérés.

Pour plus de renseignements sur ce sujet, nous renvoyons le lecteur aux travaux du groupe de travail COSOCATI[1], au sein duquel notre équipe a participé (dans l'annexe F page 179, vous trouverez le travail que j'ai effectué à proprement parlé). Pour Internet et l'*ADSL*, nous vous renvoyons au travail fait dans notre équipe par M. Antonini *et al.* [6]. Pour la récupération de blocs, nous vous renvoyons aussi au travail fait dans notre équipe par J. Jung *et al.* [58, 59].

A.3 $\mathcal{M}$PEG7

$\mathcal{M}$PEG7 désire normaliser la description de tous les documents multimédias (aussi bien les sons, les images, les séquences vidéos, tout ce qui est imaginable aujourd'hui et qui le sera demain), en vue d'en permettre l'accès et la manipulation.

[1] COdage conjoint SOurce-CAnal pour la Transmission d'Images, c'est un projet RNRT avec comme partenaires THALES, le CNES Toulouse, l'ENST Bretagne, l'ENST Paris, le L2S, France Télécom R&D et I3S.

L'étude de cette norme est lancée en 1997. Nous effectuons dans la suite un rapide tour d'horizon, en nous intéressant surtout à l'indexation de séquences vidéos.

A.3.1 Principes généraux de $\mathcal{M}$PEG7

Lors du lancement de la normalisation $\mathcal{M}$PEG7, il était nécessaire de prendre en compte l'ensemble des applications imaginables à l'heure actuelle pour les documents audiovisuels voire multimédias : images, sons... Il s'agissait de fournir un ou des langages de description, et des descriptions de bases pouvant être utilisés par des applications (recherches d'informations, composition, navigation, etc.). Comme pour $\mathcal{M}$PEG1-2 ou $\mathcal{M}$PEG4, la norme ne s'occupe pas des outils permettant la mise en place des descriptions.

Les principaux concepts de $\mathcal{M}$PEG7 sont les suivants :

- données : les informations audiovisuelles ;
- caractéristiques : caractéristiques distinctives des données signifiant quelque-chose à quelqu'un ;
- descripteurs (D) : représentation d'une caractéristique qui en définit syntaxe et sémantique (histogramme de couleur, texte de titres...) ;
- valeurs de descripteurs : instanciation d'un descripteur (le fait de créer une instance du descripteur à partir de sa classe) ;
- schémas de description (*DS*) : schéma spécifiant la structure et la sémantique des relations entre ses composants, qui peuvent être des descripteurs ou des schémas de description. Un schéma de description se différencie d'un descripteur au sens où un descripteur ne contient que des types de données basiques ;
- descriptions : association d'un schéma de description et de valeurs pour les descripteurs qu'il contient ;
- descriptions $\mathcal{M}$PEG7 codées : descriptions encodées dans un format exploitable et conforme à la norme ;
- langage de définition de description (*DDL*) : langage permettant de décrire les schémas de description (et éventuellement les descripteurs structurés), mais également d'étendre des schémas de description.

Le langage de définition de description est basé sur XML[2]

A.3.2 Synthèse et analyse de la situation actuelle

L'objectif à la base de $\mathcal{M}$PEG7 est de fournir des moyens de décrire des flux audiovisuels, mais aussi de préciser comment les décrire en terme de contenu. Les concepteurs se sont rendu compte que cela était impossible, et proposent donc un

[2] *Extensible Markup Language* ou Langage Extensible de Balisage. Il est le langage destiné à succéder à *HTML* sur le *World Wide Web*. Comme *HTML* (*Hypertext Markup Language*) c'est un langage de balisage (*markup*), c'est-à-dire un langage qui présente de l'information encadrée par des balises. Mais contrairement à *HTML*, qui présente un jeu limité de balises orientées présentation (titre, paragraphe, image, lien hypertexte, etc.), *XML* est un métalangage, qui permet d'inventer à volonté de nouvelles balises pour isoler toutes les informations élémentaires (titre du film, mouvement de la caméra, mouvement de l'objet, etc.), ou agrégats d'informations élémentaires.

langage de description (*DDL*) et un ensemble de schémas de description fournissant des exemples de manière de décrire.

L'approche générale de $\mathcal{M}$PEG7 oscille entre une volonté dirigiste (afin d'obtenir une norme utile et effectivement utilisée) traditionnelle du groupe $\mathcal{M}$PEG, et une volonté de généralité pour permettre de prendre en compte l'ensemble des applications visées. Le mode de travail, qui consiste à essayer d'affiner peu à peu des schémas de description issus d'un regroupement préalable de l'ensemble des manières de décrire proposées au début des travaux n'encourage pas vraiment à avoir une vue d'ensemble. Les descripteurs proposés dans les divers schémas de description se recoupent relativement souvent.

Les principaux problèmes découlent d'une grande difficulté à différencier ce qui est syntaxique de ce qui est sémantique, l'objectivité des uns étant la subjectivité des autres, certains descripteurs pouvant se retrouver à plusieurs niveaux.

Le mélange des genres semble à cet égard relativement dangereux, car s'appliquant à ce qui est le plus difficile à gérer, à savoir le niveau sémantique et symbolique des descriptions. Ce niveau risque pourtant d'être celui qui aura le plus d'importance en termes de recherche et de manipulation de documents audiovisuels indexés, au regard des descriptions par exemple issues du traitement du signal.

La volonté de normaliser autre chose qu'un langage de description minimal pour l'audiovisuel (c'est-à-dire permettant de lier des descripteurs à des éléments multimédias) conduit les concepteurs de $\mathcal{M}$PEG7 à essayer de fournir dans un même élan normatif à la fois une syntaxe de l'audiovisuel, une sémantique, un sens commun et des genres principaux, au risque de n'arriver à rien.

Il semble que c'est au prix d'une simplification relative de la volonté de description, c'est-à-dire en fournissant au moins un langage universel permettant de décrire des descriptions de flux audiovisuels, tenant compte de ses particularités spatio-temporelles, que les résultats pourront avoir une utilité. Alors leur utilisation dans un certain nombre d'applications permettra de faire émerger des pratiques et des genres liés à l'audiovisuel totalement indexé.

A.3.3 Les champs que nous renseignons

Le mouvement de la caméra

Avec les quatre mouvements de la caméra trouvés dans la première partie, il est possible de renseigner certains champs concernant le mouvement de la caméra. Nous déterminons les parties de la séquence où le mouvement est constant ou en accélération constante. Il faut connaître le numéro de l'image où cette partie commence et le numéro où cela finit. Grâce à *XML*, nous donnons pour chaque mouvement sa valeur.

Le mouvement des objets

Pour les objets, deux représentations sont possibles :

– soit en entourant l'objet par un rectangle minimal (boîte englobante). Dans ce cas là on transfère les coordonnées du point supérieur gauche, les grandeurs de la boîte et son mouvement.
– soit en entourant l'objet par un contour (une *spline* par exemple). On transmet ce contour pour chaque image ainsi que le mouvement de son centre de gravité.

Des informations sur l'objet

Il est possible de caractériser l'objet par sa couleur, pour cela nous archivons une signature couleur pour chaque objet que nous aurons segmenté afin d'interroger une base de données et d'obtenir ce que représente l'objet. Il est à noter que l'espace de couleur que nous utilisons (L-Cr-Cb) n'est pas le plus performant dans ce domaine. Pour cela, il sera peut-être utile de recoder les objets avec un nouvel espace de couleur ayant un signature plus discriminante (pour ces espaces, nous renvoyons le lecteur aux articles de P. Le Callet et D. Barba ou M. Carnec *et al.* [19, 20]).

Annexe B

Lumière et espaces de couleur

L'œil distingue jusqu'à 10 millions de nuances de couleurs ; le but de l'homme est de réaliser un système physique le plus proche possible de la vision humaine. Dans une première partie, nous définissons ce qu'est la lumière et en particulier la couleur, ou le sentiment de couleur.

B.1 La lumière

Sans lumière point de couleur. La lumière est probablement le phénomène naturel que nous rencontrons le plus souvent et la couleur est l'un des sujets les plus complexes qui soient - le mot lui-même pouvant prendre des sens différents suivant les contextes. La couleur fait partie de la vie professionnelle de beaucoup de gens : artistes, architectes, publicitaires, imprimeurs, et elle intervient à chaque instant de notre vie - d'éveil ou de sommeil - en définissant tout ce que nous voyons, et teintant même nos rêves. Pourtant nous ne nous demandons pas d'où vient exactement la lumière. Elle est d'un côté une onde électromagnétique, se propageant dans l'espace, se modifiant avec le temps, et de l'autre un corps. En fait, nous baignons en permanence dans un champ électromagnétique créé par la présence de particules en mouvement et chargées. Une perturbation de ce champ se propagera : c'est une onde électromagnétique. Ceci donne à ces ondes la propriété de pouvoir se déplacer dans le vide. Les ondes électromagnétiques sont caractérisées par leur longueur d'onde. Mais la lumière visible n'en est qu'un sous-ensemble réduit. Le spectre du visible, pour l'homme, s'étend à peu près de 400 nm (violet) à 700 nm (rouge).

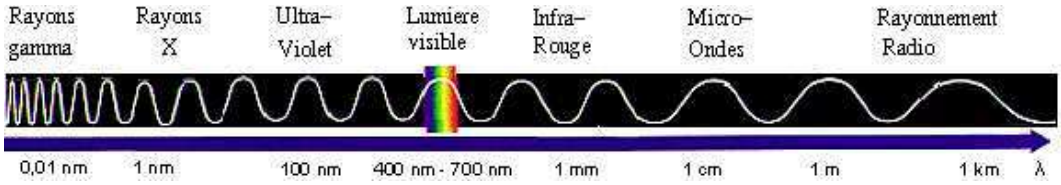


FIG. B.1 : Spectre des ondes électro-magnétiques

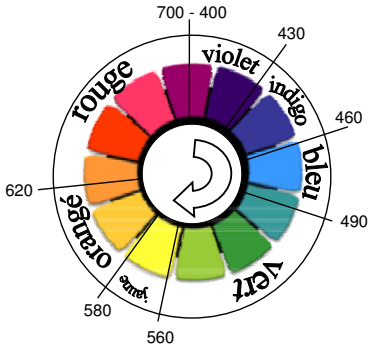


FIG. B.2 : La roue des couleurs (de 400 nm à 700 nm)

Les figures B.1 et B.2 sont prises sur :
http ://www.bioinformatics.org/oeil-couleur/dossier/lumiere.html
http ://www.ac-reims.fr/datice/HOTELLERIE/artflora/couleurs.htm respectivement.

Les travaux de Max Planck et d'Albert Einstein sur la 'lumière quantique' ont montré que l'énergie de la lumière est aussi en quelque sorte 'granuleuse' ; ce 'grain d'énergie' étant appelé un photon. Chaque photon d'un rayonnement (lumière, rayons X...) porte une quantité d'énergie caractéristique de sa fréquence ; il est une particule élémentaire de la famille des bosons, et explique les échanges d'énergie entre lumière et matière.

Ces échanges avec la matière permettent de comprendre en effet comment on peut voir certains objets. Il existe deux types d'objets :

- les producteurs de lumière, comme le Soleil, les flammes, les lampes à incandescence, etc. Ils produisent souvent de la lumière par transformation calorifique ; le mouvement perpétuel d'agitation de la matière émettant des ondes électromagnétiques. Celles-ci ne sont pas toujours visibles mais lorsque la température augmente, cela passe du rayonnement infrarouge (non visible) au spectre du visible, puis, à nouveau dans le non visible, le rayonnement ultraviolet.
- les objets qui ne sont visibles que s'ils sont éclairés. Ils diffusent dans toutes les directions la lumière qu'ils reçoivent, ce qui est appelé émission atomique ou moléculaire. En fait, un atome ou une molécule peut s'exciter par apport d'énergie extérieure (par chaleur, lumière, ou décharge électrique). Dans ce cas, un électron peut accéder à un niveau d'énergie supérieur, mais il ne tardera pas à reprendre sa place d'origine à cause de l'instabilité de cet état. Lorsqu'il reprendra sa place, il cédera de l'énergie sous forme d'un photon émis. Celui-ci portera donc une énergie proportionnelle à la fréquence de la radiation. Selon la complexité de l'atome (resp. de la molécule), l'émission lumineuse pourra être constituée d'un grand nombre de longueurs d'ondes, matérialisée par des raies dans le spectre (resp. de bandes continues). De ce fait, une couleur est définie par sa longueur d'onde, ou par un mélange de longueurs d'ondes. Par exemple, un vert 'pur' est une radiation monochromatique de longueur d'onde 530 nm, tandis que la lumière blanche est un spectre continu contenant toutes les longueurs d'ondes du domaine du visible.

À la base de la perception, nous avons un rayon lumineux. Celui-ci traversera les milieux transparents de l'œil et, du fait de leurs différents indices de réfraction, il sera réfracté plusieurs fois, la cornée et le cristallin assurant l'accommodation de l'image. La lentille convergente formée par ces milieux transparents provoque aussi une inversion de l'image sur la rétine, qui sera corrigée lors de l'interprétation cérébrale. Puis le rayon traverse la rétine, pour arriver finalement aux segments externes des photo-récepteurs, où il sera interprété. C'est le système des cônes, qui est un système à haute résolution mais dont la sensibilité est limitée, qui effectue cette interprétation. Des biologistes ont mis en évidence trois sortes de cônes dans la rétine humaine, ayant une absorption maximale, l'une dans le bleu-violet, plus exactement à 420 nanomètres, la deuxième dans le vert, à 530 nm, et la troisième à 565 nm, dans le jaune-rouge. Ces résultats ont été confirmés par l'extraction de trois sortes de pigments des cônes de rétines humaines : un pigment sensible au bleu, le deuxième sensible au vert, le troisième au rouge. Ces spécificités ont permis de classer les cônes humains en trois catégories (FIG. B.3[1]) :

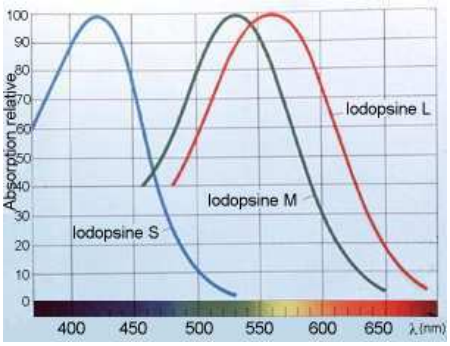


FIG. B.3 : Spectre d'absorption

- les cônes S, qui contiennent en majorité le pigment sensible au bleu ;
- les cônes M, présentant une concentration plus importante en pigments sensibles au vert ;
- les cônes L, porteurs du pigment sensible au rouge.

Les lettres conventionnelles S, M et L proviennent des mots anglais *Short*, *Medium* et *Long wavelenght*, qui correspondent respectivement aux courtes, moyennes et grandes longueurs d'ondes.

Après l'acquisition de l'image, c'est par le nerf optique que l'information est amenée au cerveau et l'on se rend compte que presque un tiers de la matière grise du cerveau traite le signal visuel : traitement de la couleur, du contour, du mouvement, de l'orientation, de la profondeur.

Il faut modéliser la représentation des couleurs que va avoir un système optique artificiel, afin d'être le plus proche possible du système optique humain. En effet, nous devons favoriser les informations que le cerveau va étudier avec précision et par contre abandonner tout ou partie des informations non traitées. Il est par exemple

[1] figure prise sur http ://daltonien.free.fr/daltonisme.php

inutile de transmettre l'image dans les rayons X, car le cerveau ne l'utiliserait pas. Cela permet, facilement, un gain de bande passante.

B.2 Les différents espaces de couleur (RVB, YUV...)

Un espace de couleur est la représentation mathématique d'une palette de couleurs ; de plus, il est un composant très important de l'image. Il existe de nombreuses méthodes de représentation des couleurs, certaines se trouvent dans les articles de R.W.G. Hunt, M. Kunt *et al.*, M. Miyahara et Y. Yoshida, N. Ohta ou G. Wyszecki et W.S. Stiles [50, 62, 72, 84, 117], qui découlent tous de l'espace de couleur Rouge - Vert - Bleu (RVB) soit après une transformation linéaire, soit non linéaire. Nous sommes dans l'optique de la segmentation, et chaque espace de couleur a des points positifs mais aussi négatifs selon le type de l'image que nous traitons. Y.L. Ohta *et al.* [85], proposent de voir les espaces de couleur les mieux adaptés (ils tentent, à chaque fois, d'être le plus proche de la vision humaine). Pour notre part, nous utilisons directement le flux $\mathcal{M}$PEG1-2, de ce fait, nous ne sommes pas libres du choix de l'espace de couleur. Les espaces de couleur que nous étudions sont :

- l'espace RVB ;
- l'espace X-Y-Z ;
- l'espace L-Cr-Cb (Luminance, Chrominance rouge, Chrominance bleue) ;
- les espaces *HSI*, *HSV* et *HLS* ;
- L'espace CMYK (*Cyan Magenta Yellow blacK*).

B.2.1 L'espace RVB

C'est l'espace de couleur standard pour les images numériques. En effet, les capteurs des caméras *CCD* (*Coupled Charge Device* soit Dispositif à Transfert de Charges) donnent un flux de sortie RVB sous la forme d'une tension électrique. Celui-ci est transformé en trois nombres décimaux compris entre zéro et un. Cela décrit quelles quantités de rouge, de vert et de bleu composent un certain pixel. Par exemple, la couleur rouge primaire est codée dans la notation RVB par 1-0-0 et le blanc par 1-1-1 (*cf.* le cube des couleur FIG. B.4[2]).

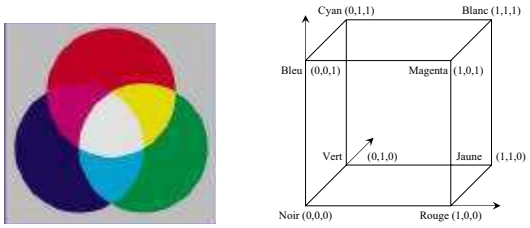


FIG. B.4 : Système additif de l'espace RVB (avec le cube des couleurs)

[2]figure prise sur http ://semsci.u-strasbg.fr/primcol.htm

Le rouge correspond à une fréquence de 700 nm, le vert de 546,1 nm et le bleu de 435,8 nm. Pour obtenir ce rouge, il faut une lampe à incandescence et un filtre rouge normalisé. Pour le vert et le bleu, ils correspondent respectivement à la raie verte et à la raie bleue de l'arc au mercure. Nous pouvons, grâce à cela, définir toute couleur C comme la somme des quantités de chaque couleur primaire (rouge, verte et bleue). Dans cette représentation, la somme de deux couleurs quelconques a pour équivalent la somme de leurs composantes primaires respectives.

B.2.2 L'espace X-Y-Z

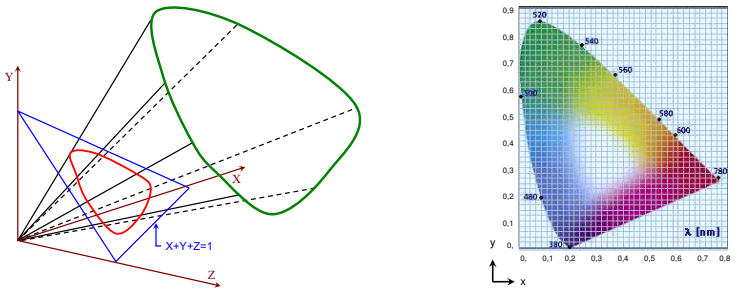
Dans cet espace, le Y représente la luminance et le X et le Z les deux chrominances, Y étant indépendant de X et Z. Le plan (X,Z) forme le plan dit de luminance nulle. Nous passons du système RVB au système XYZ par une transformation linéaire de coordonnées.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2,77 & 1,75 & 1,13 \\ 1 & 4,59 & 0,06 \\ 0 & 0,05 & 5,59 \end{pmatrix} \begin{pmatrix} R \\ V \\ B \end{pmatrix} \tag{B.1}$$

Ces primaires X,Y,Z, sont dites irréelles puisqu'elles ne correspondent pas à une lumière qu'il est possible de produire. Dans la pratique, nous effectuons un changement d'échelle en utilisant trois autres coefficients x, y, et z avec comme relation $x + y + z = 1$. La connaissance de seulement deux de ces coefficients permet de déduire le troisième.

$$x = \frac{X}{X+Y+Z}, y = \frac{Y}{X+Y+Z}, z = \frac{Z}{X+Y+Z}$$

D'où x et z donnent les informations de chrominances et y porte l'information de luminance.



(a) Espace XYZ (b) Diagramme de chromaticité (x,y)

FIG. B.5 : L'espace XYZ

Dans la FIG. B.5-b[3], la droite reliant les deux extrémités de la courbe représente la ligne des couleurs issues du mélange du bleu et du rouge. La surface coloriée

[3] figure prise sur http ://tecfa.unige.ch/~lombardf/CPTIC/couleurs/couleur_ERAG/Pages/ch8p1.htm

est appelée *Spectrum Locus* (lieu spectral) ; elle contient toutes les couleurs réelles. C'est dans cette représentation que sont estimées les couleurs complémentaires.

Pour information, une couleur est la complémentaire d'une autre si le mélange des deux donne du blanc en synthèse additive, ou du noir en synthèse soustractive. La complémentaire d'une couleur est la même dans les deux systèmes (additif ou soustractif). La complémentaire d'une couleur primaire est une couleur secondaire. Par exemple, la couleur complémentaire du bleu est le jaune ; en effet du bleu mélangé à du jaune donne du blanc en synthèse additive et du noir en synthèse soustractive.

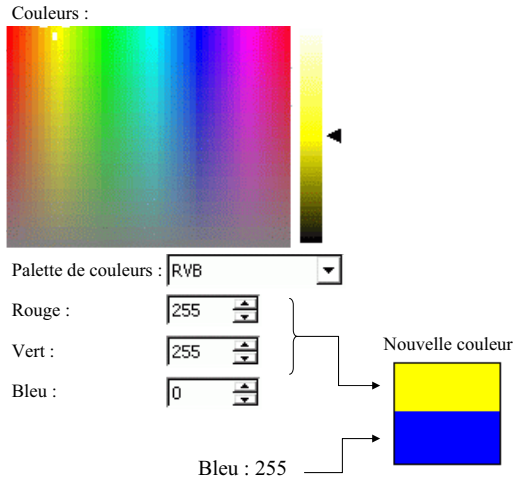


FIG. B.6 : La couleur complémentaire du bleu.

B.2.3 L'espace L-Cr-Cb

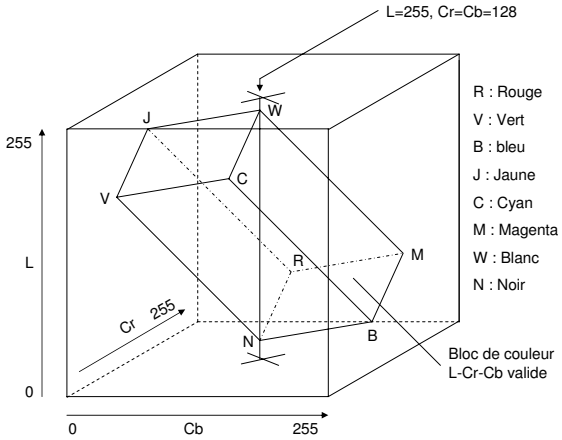


FIG. B.7 : Représentation de l'espace L-Cr-Cb

Cet espace fait partie d'un ensemble défini en 1976 par la CIE (Commission

Internationale d'Éclairage). Il est généré en transformant linéairement l'espace RVB en espace yuv puis, enfin, en passant à L-Cr-Cb par une transformation non linéaire.

$\mathcal{M}$PEG1-2 définit cinq modes de transformations (cinq recommandations) pour passer de l'espace RVB à l'espace yuv, soit cinq matrices T. Le numéro du mode est passé dans la trame à décoder, il est bien sûr nécessaire lors du décodage. La transformation suit le calcul :

$$\begin{pmatrix} y \\ u \\ v \end{pmatrix} = T. \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{B.2}$$

en posant :

$$T = \begin{pmatrix} t_{0,0} & t_{0,1} & t_{0,2} \\ t_{1,0} & t_{1,1} & t_{1,2} \\ t_{2,0} & t_{2,1} & t_{2,2} \end{pmatrix} \tag{B.3}$$

et en ayant les trois valeurs $t_{0,0}$, $t_{0,1}$ et $t_{0,2}$, une même transformée permet d'estimer les six valeurs restantes, ceci à partir du système (B.2) et en posant :

$$\begin{cases} u = \frac{0,5}{1-t_{0,2}} (B - y) \\ v = \frac{0,5}{1-t_{0,0}} (R - y) \end{cases} \tag{B.4}$$

Après estimation pour chaque recommandation, nous obtenons les matrices de transformation linéaire T :

– si la transformation suit la recommandation 709 de l'ITU-R, nous avons :

$$T = \begin{pmatrix} 0,2125 & 0,7154 & 0,0721 \\ -0,115 & -0,386 & 0,5 \\ 0,5 & -0,454 & -0,046 \end{pmatrix}$$

– si la transformation suit la recommandation FCC, nous avons :

$$T = \begin{pmatrix} 0,3 & 0,59 & 0,11 \\ -0,169 & -0,331 & 0,5 \\ 0,5 & -0,421 & -0,079 \end{pmatrix}$$

– si la transformation suit la recommandation 624-4 système B, G de l'ITU-R ou bien la transformation SMPTE 170M, nous avons :

$$T = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,169 & -0,331 & 0,5 \\ 0,5 & -0,419 & -0,081 \end{pmatrix}$$

– si la transformation suit la recommandation SMPTE 240M, nous avons :

$$T = \begin{pmatrix} 0,212 & 0,701 & 0,087 \\ -0,116 & -0,384 & 0,5 \\ 0,5 & -0,445 & -0,055 \end{pmatrix}$$

Maintenant avec les valeurs y, u et v, nous appliquons une transformation non linéaire :

$$\begin{cases} L &= \left\lceil \frac{219}{256} y + 16,5 \right\rceil \\ Cr &= \left\lceil \frac{224}{256} v + 128,5 \right\rceil \\ Cb &= \left\lceil \frac{224}{256} u + 128,5 \right\rceil \end{cases} \tag{B.5}$$

Connaissant les bornes des valeurs y, u et v et en les introduisant dans le système (B.5) nous remarquons que la luminance est comprise entre 16 et 235 et que les chrominances sont elles comprises entre 16 et 240.

L'avantage de passer de RVB à L-Cr-Cb est que l'information est dissociée en luminance, qui est la partie la plus porteuse d'informations pour notre cerveau, et en chrominances. Nous verrons dans ce qui suit, comment $\mathcal{M}$PEG1-2 utilise cette connaissance psycho-visuelle.

La vision colorée est caractérisée par trois sensations distinctes, traduites par des grandeurs subjectives qui sont la luminance, la teinte et la saturation. Le but pour un espace de couleur est d'être le plus proche de la vision humaine, afin de ne transmettre que les informations qui seront pertinentes pour le cerveau. Plusieurs mots permettent de définir une partie de la couleur :

- La luminance : elle qualifie l'impression d'intensité ou de vivacité d'une lumière ; elle est liée à la puissance du rayonnement reçue par l'œil, et bien sûr, à la sensibilité de celui-ci en fonction de la longueur d'onde.
- La teinte est la sensation colorée que nous interprétons en fonction de la longueur d'onde dominante de la radiation.
- La saturation est liée à la pureté ou la saturation de la radiation.

B.2.4 Les espaces *HSI*, *HSV* et *HLS*

Une couleur peut être vive (claire et saturée), pâle (claire et désaturée), profonde (foncée et saturée), ou rabattue (foncée et désaturée).

Ce sont ces notions qu'utilisent les espaces *HSI*, *HSV* et *HLS*.

HSI pour *Hue Saturation Intensity* (nuance, saturation et intensité), *HSV* pour *Hue Saturation Value* (nuance, saturation et luminosité), et enfin *HLS* pour *Hue Lightness Saturation* (nuance, luminance et saturation).

Ce sont des espaces souvent représentés par un cône ou un cône hexagonal. Ils traduisent mieux les composantes naturelles de la couleur, et reflètent peut-être mieux la perception humaine de la couleur. L'espace *HLS* est similaire à *HSI*. Le terme *Lightness* (luminance) est similaire au terme *Intensity* (intensité).

La différence entre *HSI* et *HSV* se situe dans l'estimation de la composante contraste (I ou V), qui détermine sa distribution et sa dynamique. L'espace *HSI* est utilisé lorsque des opérations sont faites sur les images (comme la convolution, les histogrammes...). En effet, les manipulations sur le contraste I ont un impact identique sur chaque composante de l'espace RVB. L'espace *HSV*, par contre, est préféré pour les manipulations sur la nuance (H) et sur la saturation (S).

L'intérêt de ces espaces est que ces composantes sont très proches de la perception humaine des couleurs. Pour passer de RVB à *HSV*, *HSI*, ou *HLS*, il faut effectuer une transformation non linéaire (article de J.R. Smith [101]).

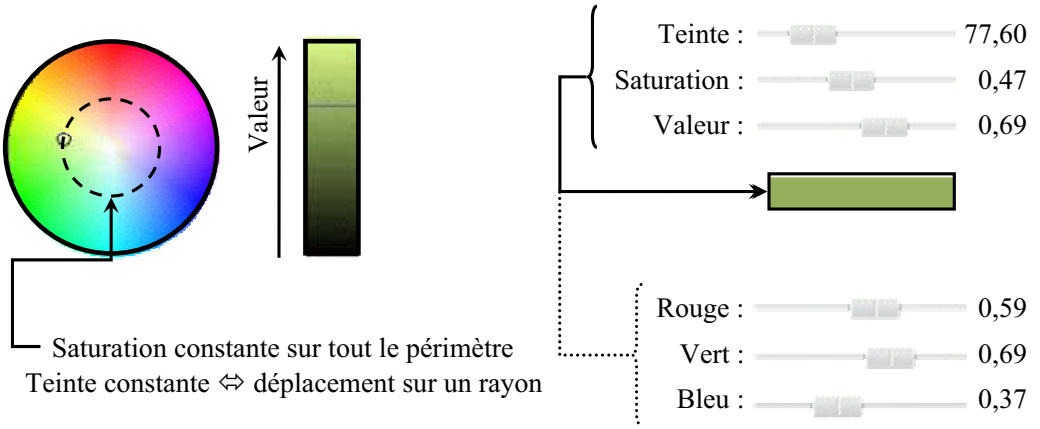


FIG. B.8 : Espace de couleur plus proche de la vision humaine (correspondance d'une couleur dans cet espace et dans l'espace RVB).

Les composantes des espaces *HLS* et *HSV* ne sont pas complètement identiques. La composante Nuance dans les deux espaces est composée de mesures angulaires, analogues à la position autour du disque de couleurs. Une nuance égale à zéro indique la couleur rouge, 120 pour le vert et 240 pour le bleu.

La composante saturation dans les deux espaces donne l'intensité de la couleur. Une saturation égale à zéro indique que l'image est en niveaux de gris.

La composante Luminosité pour *HSV* et Luminance pour *HLS* sont à peu près identiques. Une valeur égale à zéro représente la couleur noire ; *a contrario*, une valeur maximale donne une couleur la plus lumineuse possible. Dans l'espace *HLS*, une luminosité située à son maximum correspond à la couleur blanche.

Des améliorations peuvent toujours être apportées à ces espaces pour obtenir un rendu des images encore plus proche de la vision humaine, pour cela nous vous renvoyons aux articles de P. Le Callet et D. Barba ou M. Carnec *et al.* [19, 20]. Mais pour valider un nouvel espace, il y a un problème de subjectivité (article de H. Senane *et al.* [98]). Il faudrait pourvoir normaliser une méthode objective pour quantifier sa proximité avec la vision (article de A.M. Rohaly *et al.* [93]), mais cela parait très difficile. En effet, comme nous disions dans l'introduction, chacun a une vision différente...

B.3 L'espace CMYK (*Cyan Magenta Yellow blacK*)

Cet espace, en français cyan, magenta, jaune et noir, est utilisé généralement dans les systèmes d'imprimantes couleurs. Il est dépendant de la machine et de plus est de nature soustractive (contrairement au RVB additif qui est pris généralement).

Dans cet espace, Cyan, Magenta et Jaune sont trois couleurs primaires, et rouge, vert et bleu sont trois couleurs secondaires. Théoriquement, la couleur noire n'est pas nécessaire, mais avec le mélange des encres cyan, magenta et jaune, nous obtenons un marron foncé et non pas le noir.

Annexe C

Estimation du vecteur directeur du mouvement

Nous considérons le repère orthonormé et nous regardons pour chaque mouvement du point P, son déplacement sur la rétine.

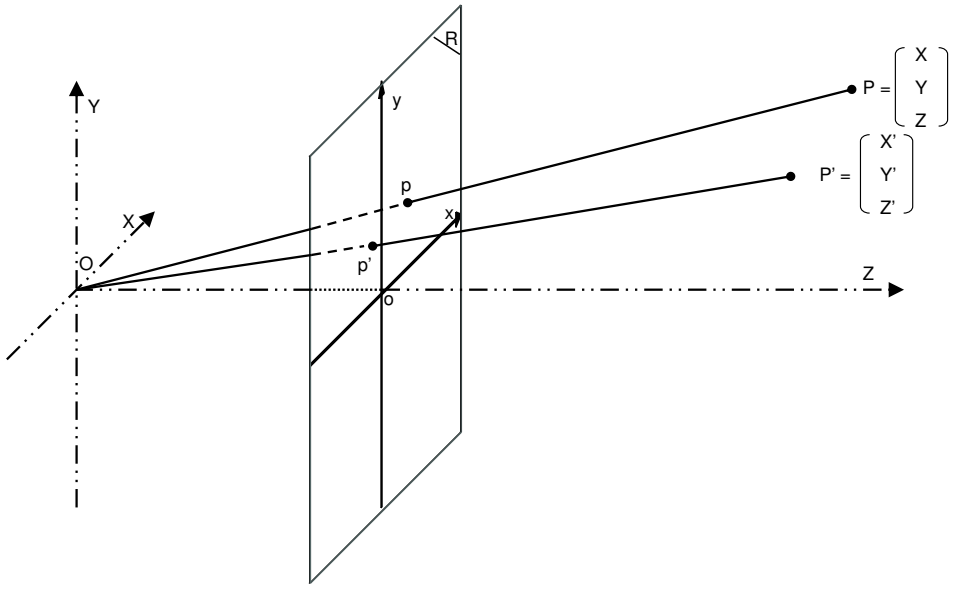


FIG. C.1 : Repère orthonormé de notre caméra

Nous avons le point initial P et le point après mouvement P' :

$$P = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \text{ et } P' = \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$

C.1 *Track*

Nous effectuons la translation du point P selon l'axe des X. Nous notons cette translation T_x, d'où :

$$P' = \begin{pmatrix} X + T_x \\ Y \\ Z \end{pmatrix}$$

Nous projetons P et P' sur la rétine en p et p' respectivement, nous obtenons :

$$p = \begin{pmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \end{pmatrix}$$

$$p' = \begin{pmatrix} \frac{f(T_x+X)}{Z} \\ \frac{fY}{Z} \end{pmatrix}$$

d'où le vecteur déplacement sur la rétine :

$$\overrightarrow{pp'} = \begin{pmatrix} \frac{fT_x}{Z} \\ 0 \end{pmatrix} \tag{C.1}$$

C.2 *Boom*

Nous avons uniquement un mouvement de translation sur l'axe des Y. Par le même *modus operandi* qu'au paragraphe précédent, nous obtenons directement le résultat :

$$\overrightarrow{pp'} = \begin{pmatrix} 0 \\ \frac{fT_y}{Z} \end{pmatrix} \tag{C.2}$$

C.3 *Dooly*

Nous effectuons la translation du point P selon l'axe des Z. Nous notons cette translation T_z, d'où :

$$P' = \begin{pmatrix} X \\ Y \\ Z + T_z \end{pmatrix}$$

Nous projetons P' sur la rétine p'. Nous obtenons :

$$p' = \begin{pmatrix} \frac{fX}{T_z+Z} \\ \frac{fY}{T_z+Z} \end{pmatrix}$$

d'où le vecteur déplacement :

$$\overrightarrow{pp'} = \begin{pmatrix} \frac{fX}{T_z+Z} - \frac{fX}{Z} \\ \frac{fY}{T_z+Z} - \frac{fY}{Z} \end{pmatrix}$$

Transformons dans le repère de l'image l'abscisse X et l'ordonnée Y :

$$X = \frac{xZ}{f} \text{ et } Y = \frac{yZ}{f}$$

d'où :

$$\overrightarrow{pp'} = \begin{pmatrix} -\frac{xT_z}{T_z+Z} \\ -\frac{yT_z}{T_z+Z} \end{pmatrix}$$

Si nous supposons que T_z est négligeable devant Z, nous obtenons :

$$\overrightarrow{pp'} \approx \begin{pmatrix} -\frac{xT_z}{Z} \\ -\frac{yT_z}{Z} \end{pmatrix} \tag{C.3}$$

C.4 *Tilt*

Nous effectuons la rotation R_x selon l'axe X, nous appliquons au point P une matrice de rotation rot_x :

$$rot_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(R_x) & -\sin(R_x) \\ 0 & \sin(R_x) & \cos(R_x) \end{pmatrix}$$

Nous obtenons :

$$P' = \begin{pmatrix} X \\ \cos(R_x).Y - \sin(R_x).Z \\ \sin(R_x).Y + \cos(R_x).Z \end{pmatrix}$$

Nous projetons P' sur la rétine, nous obtenons :

$$p' = \begin{pmatrix} \frac{fX}{\sin(R_x).Y+\cos(R_x).Z} \\ \frac{f[\cos(R_x).Y-\sin(R_x).Z]}{\sin(R_x).Y+\cos(R_x).Z} \end{pmatrix}$$

D'où le vecteur déplacement :

$$\overrightarrow{pp'} = \begin{pmatrix} \frac{fX}{\sin(R_x).Y+\cos(R_x).Z} - \frac{fX}{Z} \\ \frac{f[\cos(R_x).Y-\sin(R_x).Z]}{\sin(R_x).Y+\cos(R_x).Z} - \frac{fY}{Z} \end{pmatrix}$$

Ce qui donne, en transformant dans le repère de l'image :

$$\overrightarrow{pp'} = \begin{pmatrix} -\frac{x[-f+\sin(R_x).y+\cos(R_x).f]}{\sin(R_x).y+\cos(R_x).f} \\ -\frac{\sin(R_x).\left(f^2+y^2\right)}{\sin(R_x).y+\cos(R_x).f} \end{pmatrix}$$

Nous sommes dans le cas où R_x est un angle très faible, nous pouvons utiliser les développements limités :

$$\overrightarrow{pp'} \approx \begin{pmatrix} -\frac{x.y.R_x}{f} \\ -\frac{R_x.\left(f^2+y^2\right)}{f} \end{pmatrix} \tag{C.4}$$

C.5 *Pan*

Ici, pour la rotation R_y selon l'axe Y, nous appliquons au point P une matrice de rotation rot_y :

$$rot_y = \begin{pmatrix} \cos\left(R_y\right) & 0 & \sin\left(R_y\right) \\ 0 & 1 & 0 \\ -\sin\left(R_y\right) & 0 & \cos\left(R_y\right) \end{pmatrix}$$

En le projetant sur la rétine, nous obtenons :

$$\overrightarrow{pp'} = \begin{pmatrix} \frac{f\left[\cos\left(R_y\right).X+\sin\left(R_y\right).Z\right]}{-\sin\left(R_y\right).X+\cos\left(R_y\right).Z} - \frac{fX}{Z} \\ \frac{fY}{-\sin\left(R_y\right).X+\cos\left(R_y\right).Z} - \frac{fY}{Z} \end{pmatrix}$$

De la même manière que précédemment, nous obtenons :

$$\overrightarrow{pp'} \approx \begin{pmatrix} \frac{R_y.\left(f^2+x^2\right)}{f} - \\ \frac{x.y.R_y}{f} \end{pmatrix} \tag{C.5}$$

C.6 La rotation axiale

Ici, pour la rotation R_z selon l'axe Z, nous appliquons au point P une matrice de rotation rot_z :

$$rot_z = \begin{pmatrix} \cos\left(R_z\right) & -\sin\left(R_z\right) & 0 \\ \sin\left(R_z\right) & \cos\left(R_z\right) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

D'où :

$$P' = \begin{pmatrix} \cos(R_z).X - \sin(R_z).Y \\ \sin(R_z).X + \cos(R_z).Y \\ Z \end{pmatrix}$$

En le projetant sur la rétine, nous obtenons :

$$\overrightarrow{pp'} = \begin{pmatrix} \frac{f[\cos(R_z).X - \sin(R_z).Y]}{Z} - \frac{fX}{Z} \\ \frac{f[\sin(R_z).X + \cos(R_z).Y]}{Z} - \frac{fY}{Z} \end{pmatrix}$$

Ce qui donne :

$$\overrightarrow{pp'} = \begin{pmatrix} \cos(R_z).x - \sin(R_z).y - x \\ \sin(R_z).x + \cos(R_z).y - y \end{pmatrix}$$

Nous sommes dans le cas où R_z est un angle très faible, nous pouvons utiliser les développements limités :

$$\overrightarrow{pp'} \approx \begin{pmatrix} -y.R_z \\ x.R_z \end{pmatrix} \tag{C.6}$$

Annexe D

Exemples d'estimation du mouvement de la caméra

D.1 Séquence Stefan Edberg (séquence donnée par le *COST* 211[1])

Les courbes du *pan* et de la moyenne des valeurs absolues *DC* des images d'erreur se trouvent à partir de la page 65.

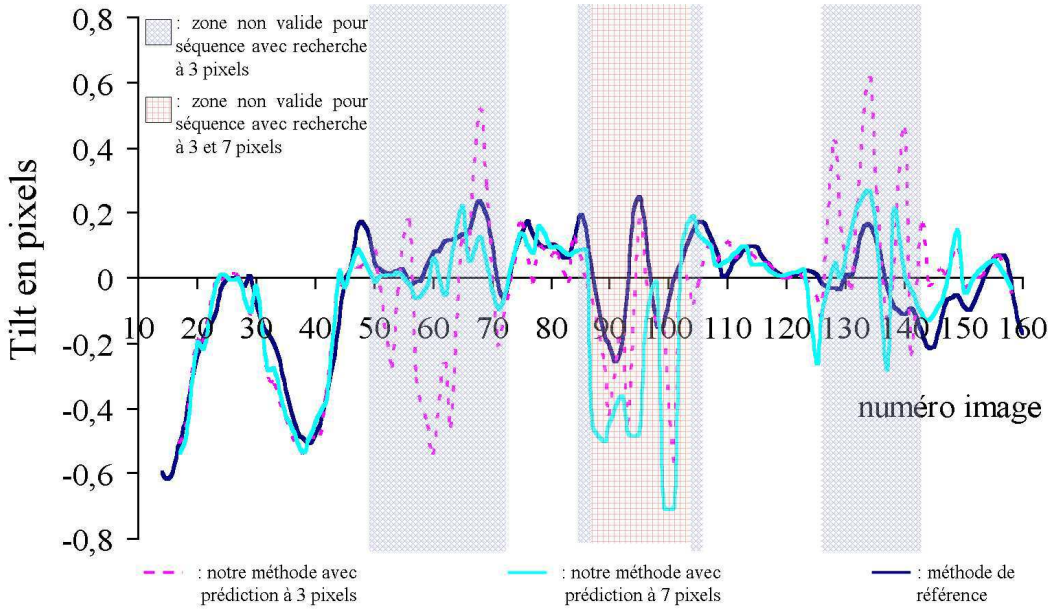


FIG. D.1 : Comparaisons entre notre méthode (avec deux flux $\mathcal{M}$PEG1-2 : prédiction à 3 pixels et à 7 pixels) et la méthode de référence : estimation de T_y

[1] *The European COST 211 Group - Research on Redundancy Reduction Techniques and Content Analysis for Multimedia Services*

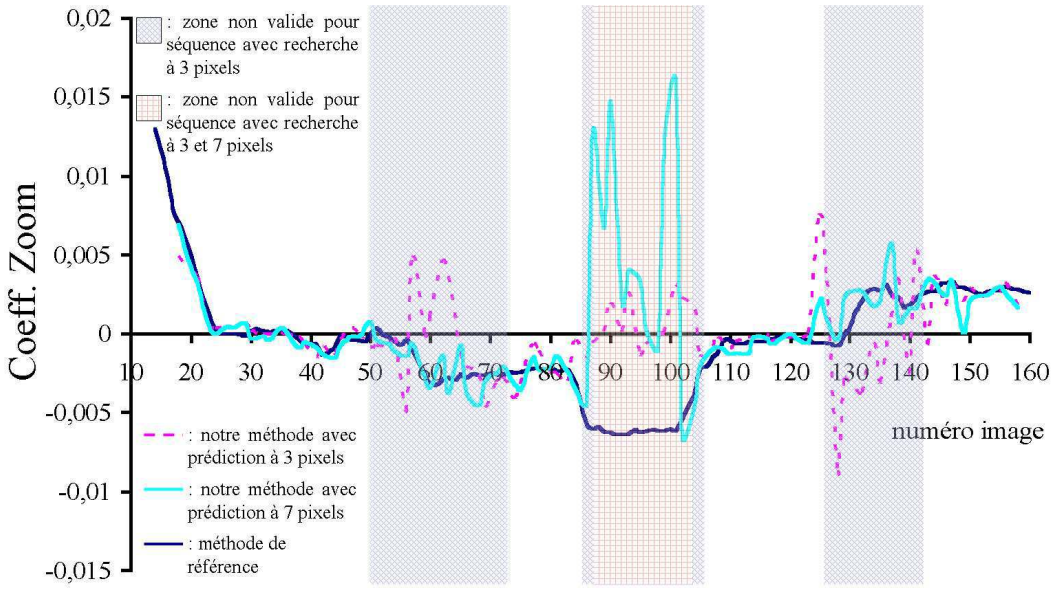


FIG. D.2 : Comparaisons entre notre méthode (avec deux flux $\mathcal{M}$PEG1-2 : prédiction à 3 pixels et à 7 pixels) et la méthode de référence : estimation de τ_z

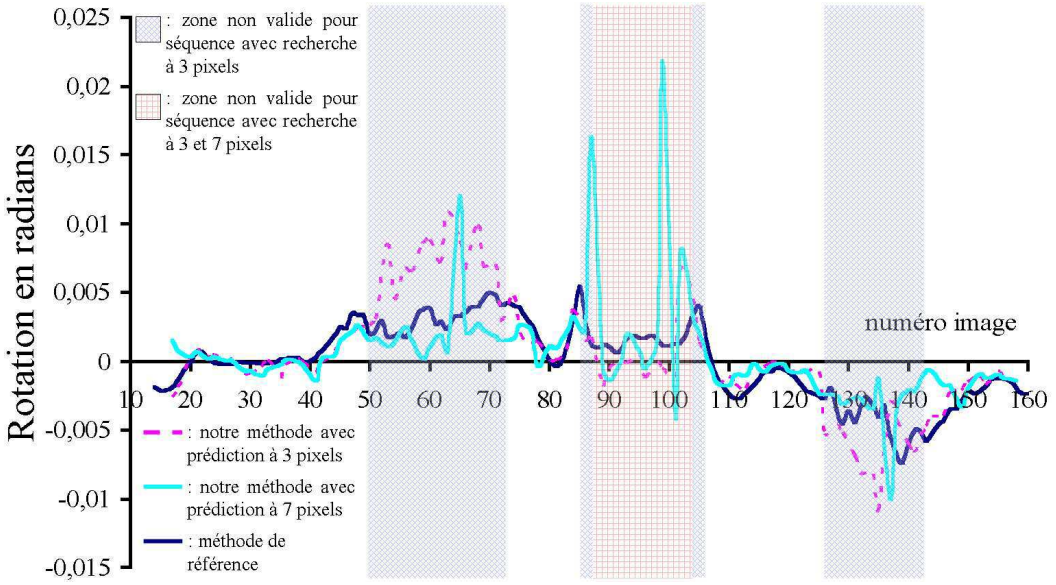


FIG. D.3 : Comparaisons entre notre méthode (avec deux flux $\mathcal{M}$PEG1-2 : prédiction à 3 pixels et à 7 pixels) et la méthode de référence : estimation de R_z

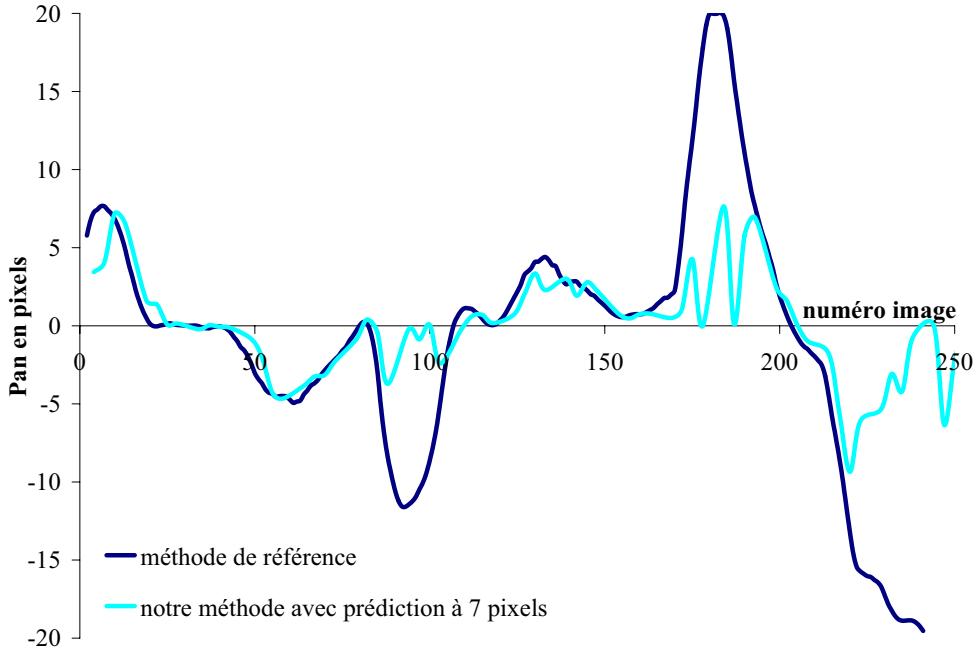


FIG. D.4 : Comparaisons entre notre méthode cette fois-ci sur la séquence de l'image 0 à 250 (avec deux flux $\mathcal{M}$PEG1-2 : prédiction à 7 pixels) et la méthode de référence : estimation de T_x

D.2 Sur la séquence *Flower-Garden*

La courbe du *pan* se trouve à la page 68.

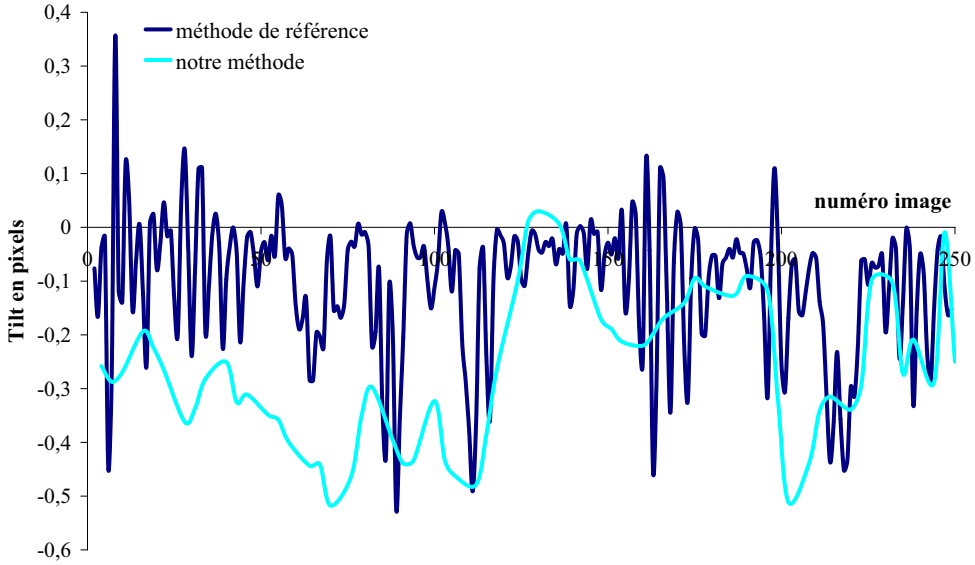


FIG. D.5 : Comparaisons entre notre méthode (avec un flux $\mathcal{M}$PEG1-2 : prédiction à 7 pixels) et la méthode de référence : estimation de T_y

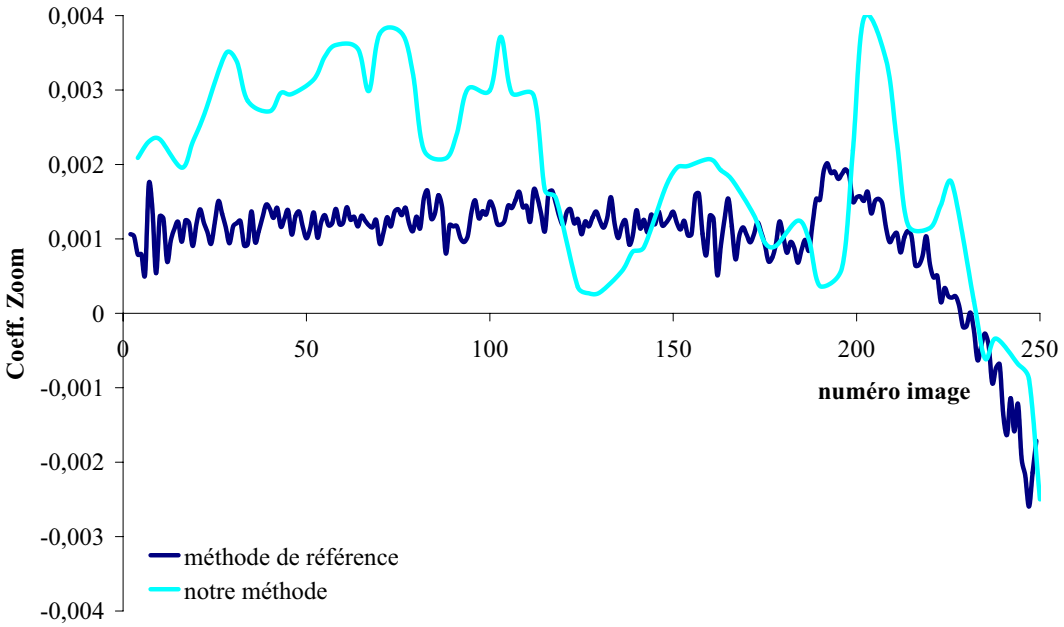


FIG. D.6 : Comparaisons entre notre méthode (avec un flux $\mathcal{M}$PEG1-2 : prédiction à 7 pixels) et la méthode de référence : estimation de τ_z

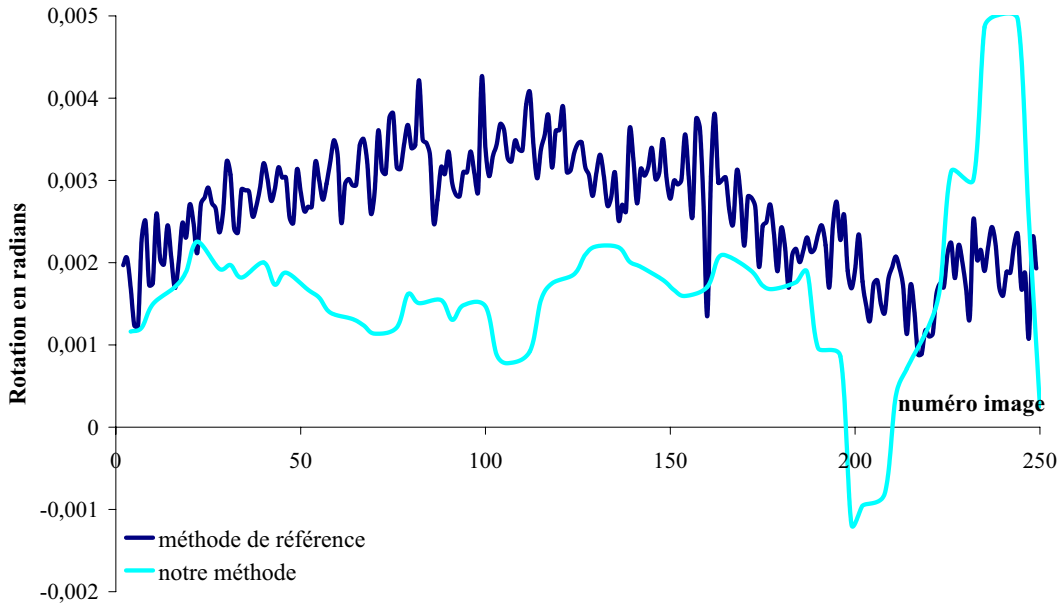


FIG. D.7 : Comparaisons entre notre méthode (avec un flux $\mathcal{M}$PEG1-2 : prédiction à 7 pixels) et la méthode de référence : estimation de R_z

D.3 Sur la séquence Bus

Les courbes du *pan*, du zoom et de la moyenne des valeurs absolues *DC* des images d'erreur se trouvent à partir de la page 69.

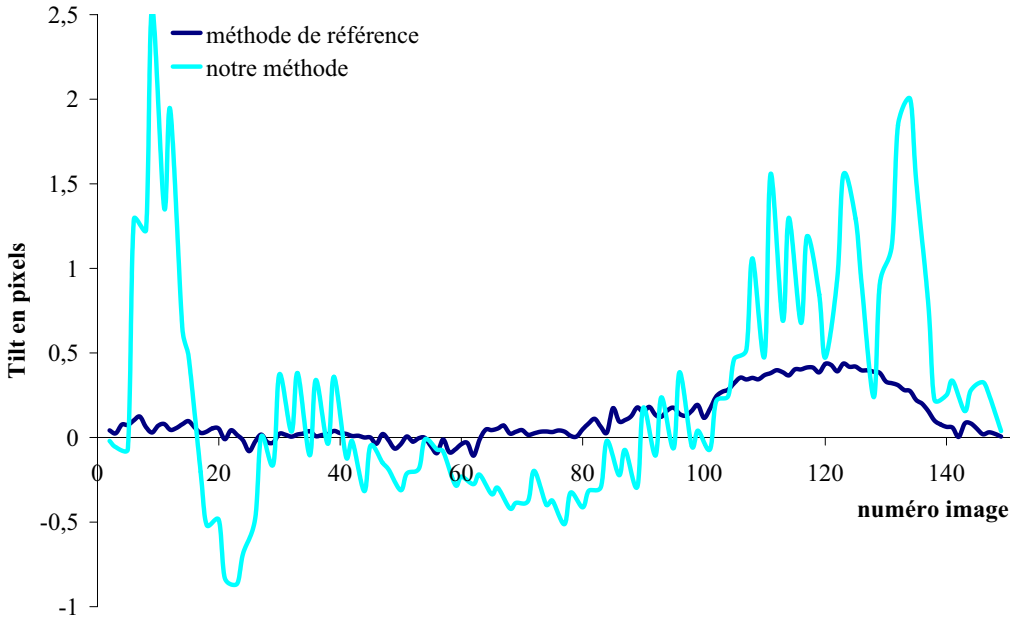


FIG. D.8 : Comparaisons entre notre méthode (avec un flux $\mathcal{M}$PEG1-2 : prédiction à 7 pixels) et la méthode de référence : estimation de T_y

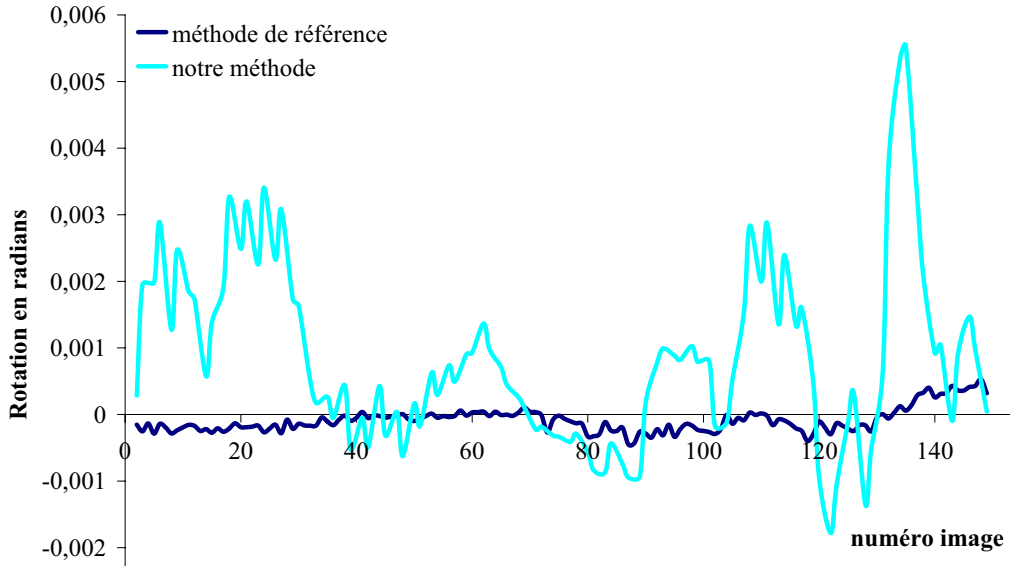


FIG. D.9 : Comparaisons entre notre méthode (avec un flux $\mathcal{M}$PEG1-2 : prédiction à 7 pixels) et la méthode de référence : estimation de R_z

D.4 Sur une séquence synthétique

Ici, entre chaque image, le *Pan* est proche de -1 pixel, le *Tilt* est proche de 1 pixel, la Rotation est proche de $-0,01$ radian et le Zoom est proche de $-0,03$. Les résultats de nos estimations sont proches de la méthode de référence. En effet, notre *Pan* et notre *Tilt* estimés sont à $\pm 0,3$ pixel de l'estimation avec la méthode de référence. Il faut garder à l'esprit que pour $\mathcal{M}$PEG1-2 la recherche du pixel le plus proche se fait avec une précision d'un demi pixel. En ce qui concerne le Zoom, le résultat est proche à $\pm 0,000\,37$, ce qui fait que sur une image de 400×400 pixels, le décalage maximal qu'il peut y avoir au bord de l'image est de $\pm 0,13$ pixel. Enfin, pour la Rotation, le résultat est proche à $\pm 0,001\,167$, ce qui fait que nous avons un décalage maximal, sur le bord de l'image, de $\pm 0,23$ pixel.

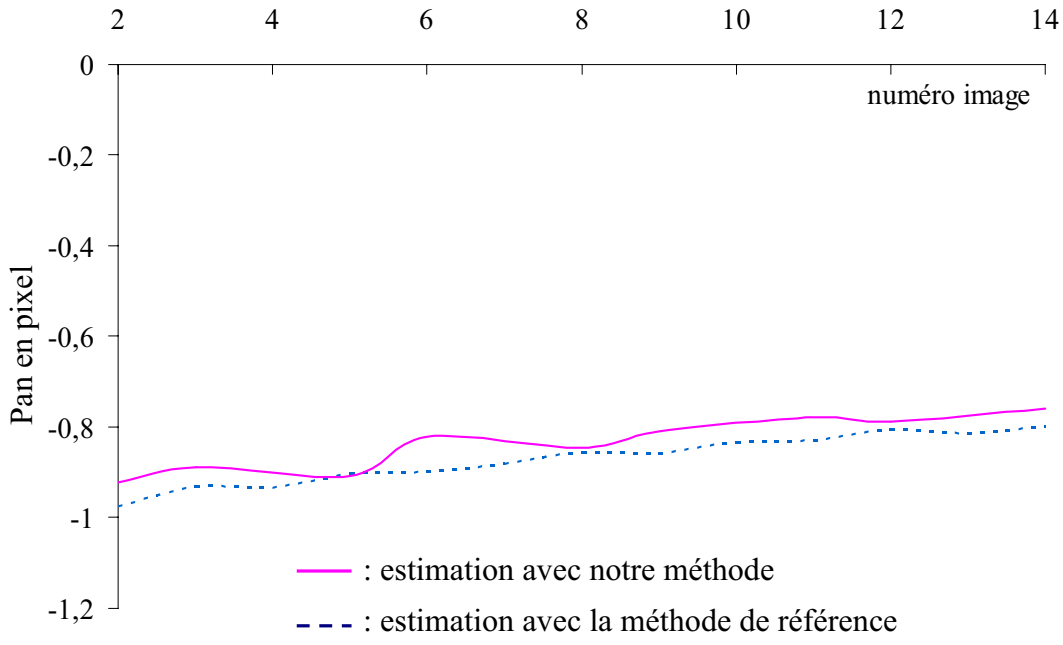


FIG. D.10 : Comparaisons entre notre méthode et la méthode de référence : estimation de T_x

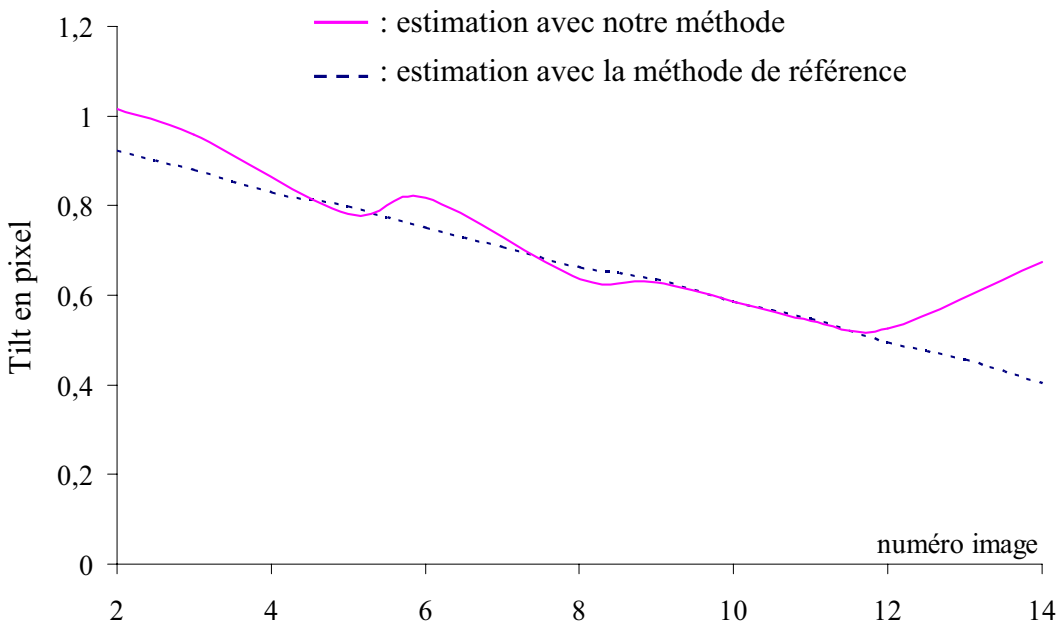


FIG. D.11 : Comparaisons entre notre méthode et la méthode de référence : estimation de T_y

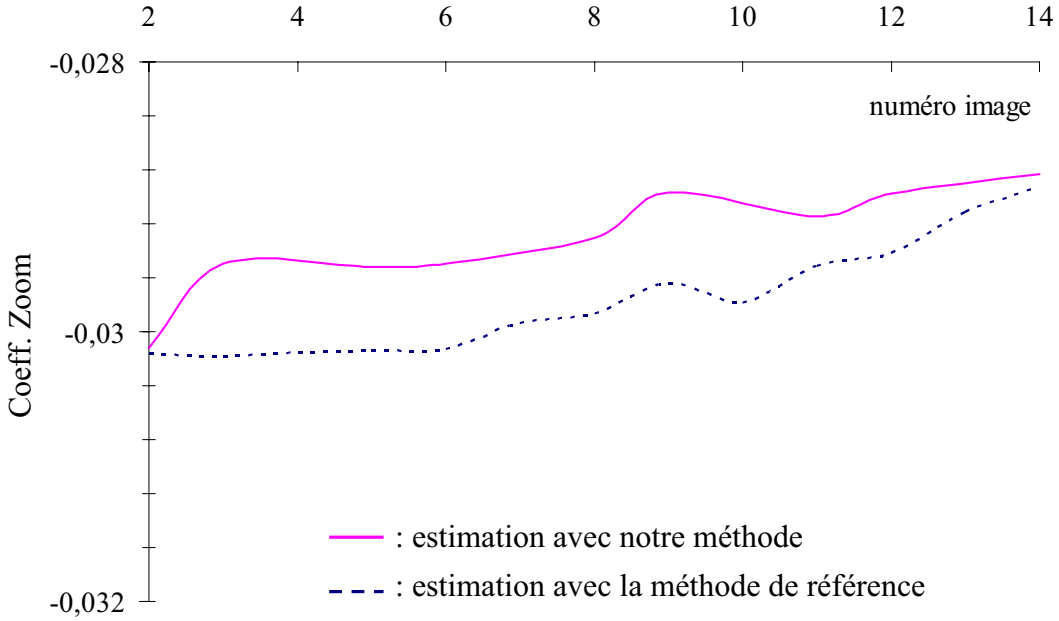


FIG. D.12 : Comparaisons entre notre méthode et la méthode de référence : estimation de τ_z

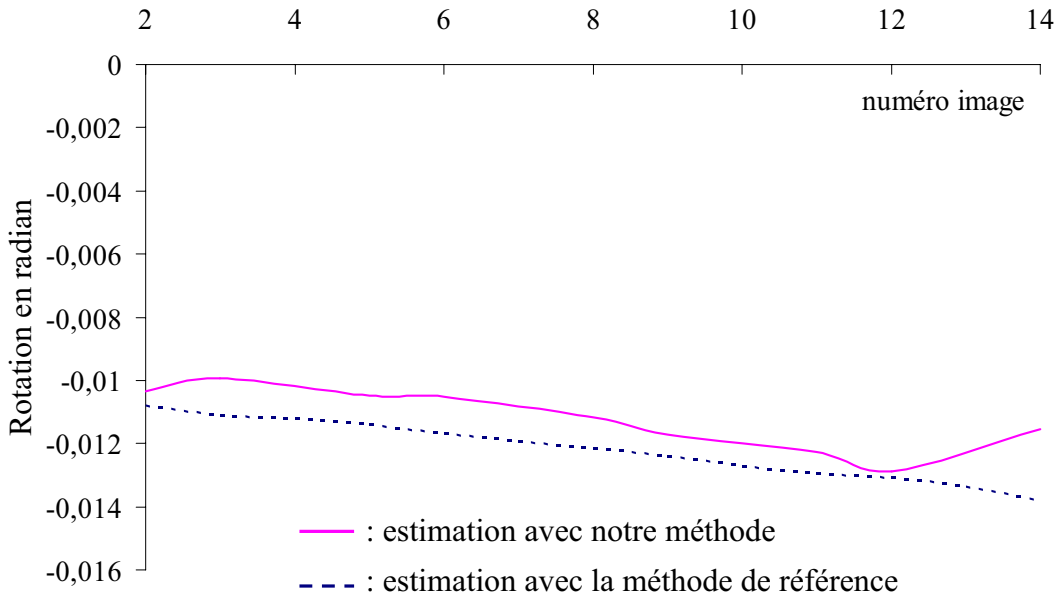


FIG. D.13 : Comparaisons entre notre méthode et la méthode de référence : estimation de R_z

Annexe E

Construction d'une *spline*

Il est possible d'interpoler une courbe par des courbes de Bézier, mais plus nous augmentons la complexité de cette courbe, plus nous augmentons le degré des courbes et par la même le coût de calcul (le degré peut en effet être supérieur à dix, ce qui est rédhibitoire en temps de calcul, même si nous sommes à l'époque des ordinateurs sur-puissants). Pour simplifier le problème, nous pouvons faire appel aux courbes de Bézier composites (polynomiales par morceaux) ou *splines*. Dans ce cas, la courbure peut être obtenue très rapidement et très précisément.

Les *cubic B-splines* ou courbes graphiques modifiables par points de contrôle furent découvertes assez tôt, dès le XIX$^{\text{ème}}$ siècle par N. Lobachevsky ; elles furent construites comme des convolutions de certaines distributions de probabilités. En 1946, I.J. Schoenberg utilisa les *B-splines* pour un lissage de données statistiques. Avec son article [97], débuta la théorie moderne de l'approximation par *B-splines*. Nous vous renvoyons aussi aux articles de R.H. Bartels *et al.* ou J.D. Foley *et al.* [10, 38].

Le but est de modéliser l'équation ou les équations qui régissent une tige flexible (ou latte) qui doit passer par plusieurs points de contrôle appelés canards. La courbe résultante est toujours très lisse. Cela suit le modèle mathématique de la minimisation d'une énergie élastique de tension E :

$$E = \int_{\Gamma(\tau)} \left[\kappa\left(x\right)\right]^2 dx \tag{E.1}$$

avec x, décrivant les points sur le contour à l'instant τ et κ sa courbure. Pour cela nous regardons du côté des arcs de Bézier de degré fixe.

Une *B-spline* peut être considérée comme une courbe complexe composée d'un assemblage de plusieurs arcs de Bézier de degré fixe. Dans le livre de G. Farin [36], pour approcher la courbe, l'auteur utilise n courbes de Bézier de degré fixe m reliées entre elles et C^{m-1} continues qui interpolent $(n+1)$ points Q_i.

Si nous considérons $m = 3$, l'auteur définit un arc *B-spline* S_i avec des points de contrôle Q_i par :

$$S_i\left(t\right) = Q_{i-1}B^4_{i-3}\left(t\right) + Q_iB^4_{i-2}\left(t\right) + Q_{i+1}B^4_{i-1}\left(t\right) + Q_{i+2}B^4_i\left(t\right) \tag{E.2}$$

avec les B_i (fonction de pondération des points de contrôle de la courbe) définis comme suit :

$$\begin{aligned}
B_i^1(t) &= \begin{cases} 1 & si \\ 0 & sinon \end{cases} \quad t \in [t_i, t_{i+1}] \\
B_i^2(t) &= \frac{t-t_i}{t_{i+1}-t_i} B_i^1(t) + \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} B_{i+1}^1(t) \\
B_i^3(t) &= \frac{t-t_i}{t_{i+2}-t_i} B_i^2(t) + \frac{t_{i+3}-t}{t_{i+3}-t_{i+1}} B_{i+1}^2(t) \\
B_i^4(t) &= \frac{t-t_i}{t_{i+3}-t_i} B_i^3(t) + \frac{t_{i+4}-t}{t_{i+4}-t_{i+1}} B_{i+1}^3(t)
\end{aligned} \tag{E.3}$$

Les arcs de Bézier successifs partagent 3 points de contrôle Q_i (FIG. E.1).

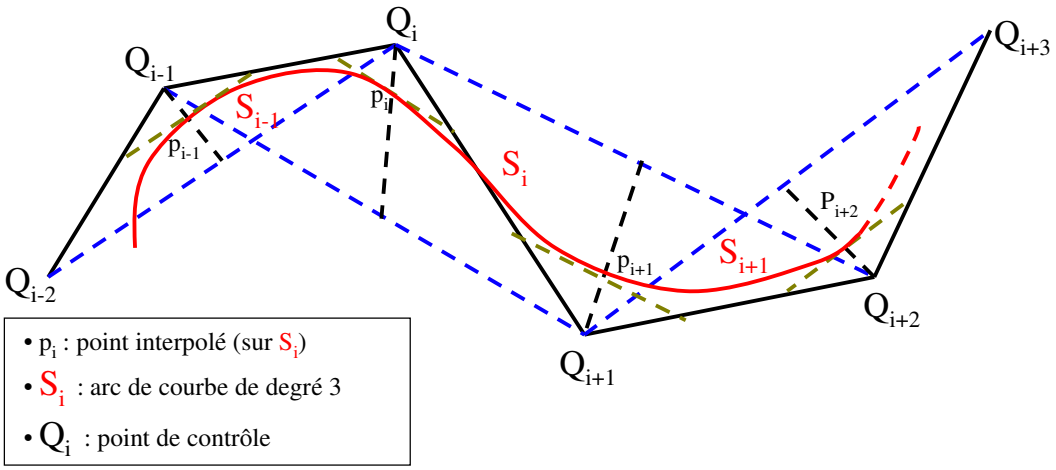


FIG. E.1 : Détails d'un arc de *B-Spline*

Sur la figure E.2 page ci-contre, nous voyons les haubans (segments de droite $Q_s P_t$) sur une *spline* réelle.

En mettant l'équation (E.2) sous forme matricielle, on obtient :

$$S_i(t) = \begin{pmatrix} 1 & t & t^2 & t^3 \end{pmatrix} . M_i . \begin{pmatrix} Q_{i-1} \\ Q_i \\ Q_{i+1} \\ Q_{i+2} \end{pmatrix} \tag{E.4}$$

En supposant que les t_i sont équidistants[1], il utilise les *B-Splines Uniformes* appelées *BSU* ce qui évite, en premier lieu, avec l'utilisation de l'invariance des courbes de Bézier par une transformation affine, de considérer l'intervalle $[t_i, t_{i+1}]$, mais au contraire l'intervalle $[0, 1]$. En second lieu, les B_i^{m+1} sont identiques d'un arc à l'autre. Dans ce cas là et après estimation, M_i devient indépendante de l'arc, ce qui permet d'avoir quel que soit i une matrice M_i unique qui sera notée M :

[1]La longueur de l'arc entre deux points de jonctions est la même quels que soient ces points, d'où $\int_{t=0}^{t=1} S_i(t).dt$ est constant quel que soit i.

$$M = \frac{1}{6}\begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \tag{E.5}$$

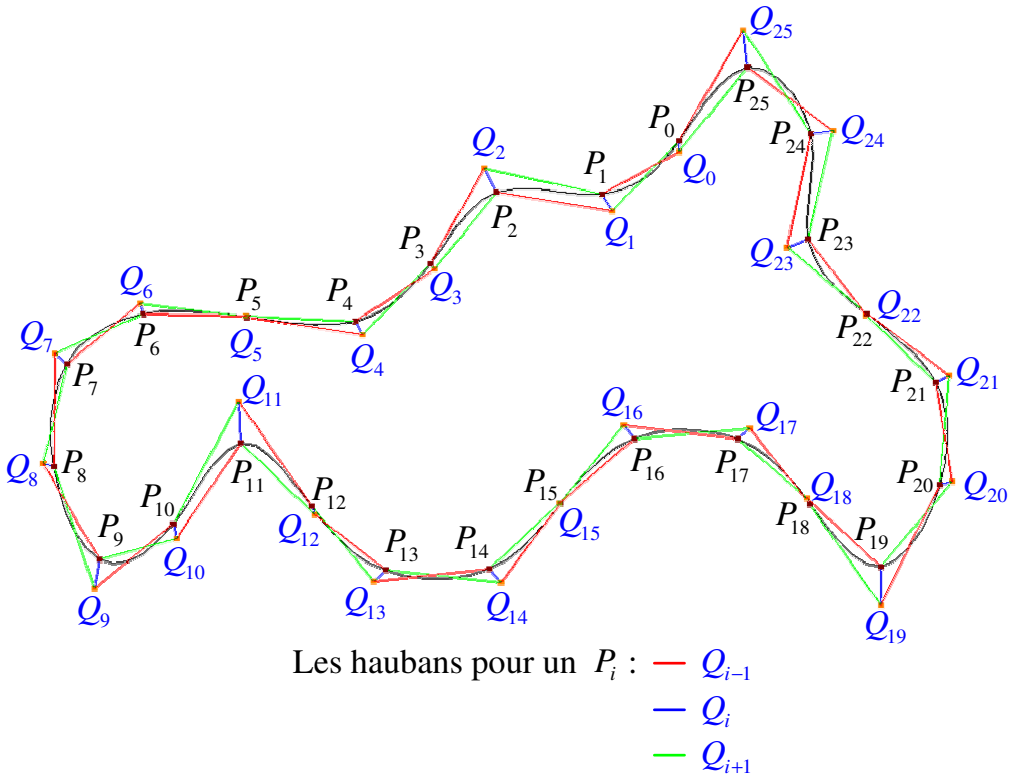


FIG. E.2 : Représentation d'une *B-Spline* fermée avec ses points Q_i et les haubans qui permettent de trouver la courbure

Il est maintenant possible d'interpoler tous les points qui vont se trouver sur la courbe. Chaque point p_i (nœud ou point de jonction), à interpoler, correspond à $S_i(t_i)$, d'où en remplaçant t par t_i dans l'équation (E.2) page 175, la position des points à interpoler peut être déduite :

$$S_i(t_i) = p_i = \frac{1}{6}\left(Q_{i-1} + 4Q_i + Q_{i+1}\right) \tag{E.6}$$

d'où :

$$S_i(0) = P_i \text{ et } S_i(1) = S_{i+1}(0) = P_{i+1}$$

Les points de contrôle P_i sont souvent nommés points de jonctions. La tangente au point p_i (voir la représentation des vecteurs tangents et normaux sur une spline réelle : FIG. E.3 page suivante) a pour équation :

$$\frac{\partial S_i\left(t_i\right)}{\partial t} = \frac{1}{2}\left(Q_i - Q_{i-1}\right) \tag{E.7}$$

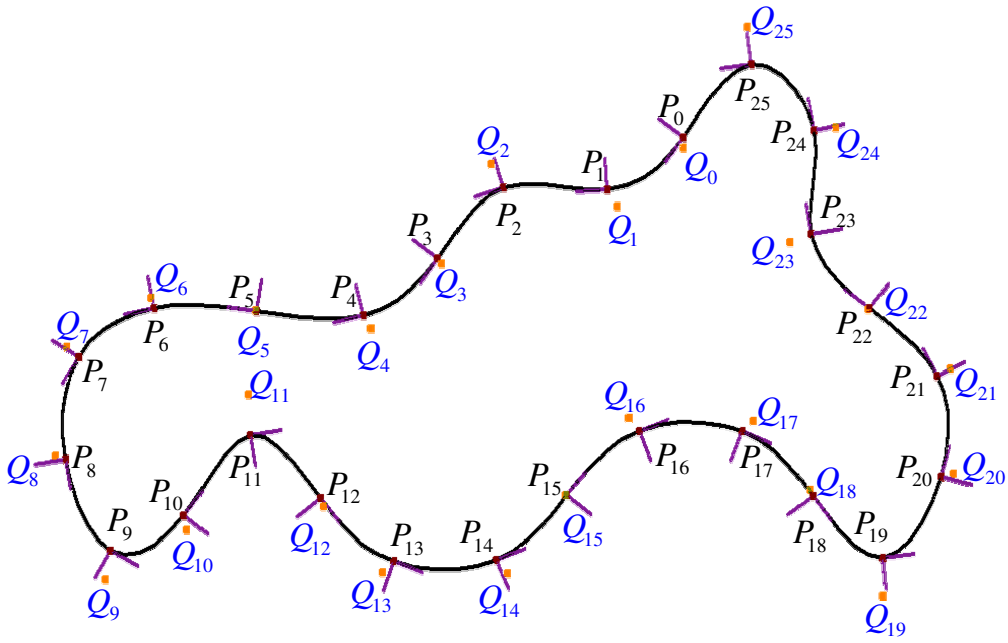


FIG. E.3 : Les vecteurs normaux et tangents en chaque nœud de la *B-Spline*

De plus, l'utilisation des *BSU* permet de conserver la régularité C^2 du contour et surtout donne directement le rayon de courbure κ au point désiré.

Annexe F

Travaux annexes

Au cours de ma thèse, avec M. Antonini, M. Barlaud, C. Boin et T. Makram, nous avons participé au projet COSOCATI[1], dans le but d'étudier la transmission de vidéos à bas débits sur des canaux radio-mobiles dans le cadre de la future norme *UMTS*.

Nos travaux ont eu pour objectif la simulation de la transmission de séquences vidéos, compressées par l'algorithme $\mathcal{M}$PEG1-2, sur des canaux radio-mobiles. En effet, la propagation des ondes radios dans l'environnement est soumise à de nombreuses perturbations ce qui affecte les téléphones portables. En effet, le signal émis peut être réfléchi sur différents obstacles, stoppé ou seulement atténué. Ainsi, la propagation peut être décrite par un nombre de trajets multiples. Dans le cas d'un canal de transmission radio-mobile, ces déformations sont généralement modélisées par un bruit de Rayleigh. La prise en compte d'un certain nombres de phénomènes physiques comme l'évanouissement ou le déphasage rend le modèle plus ou moins complexe.

Dans ce contexte, il faut analyser d'une part la structure du flux $\mathcal{M}$PEG1-2 et évaluer l'importance de chaque paquet transmis. D'autre part, il faut connaître le milieu de propagation des ondes radio.

Mon travail au sein du groupe a été la présentation du flux $\mathcal{M}$PEG1-2 et son découpage en vue de son altération (par ajout de bruit).

Algorithme mis en œuvre

Deux approches ont été mises en œuvre :

* découpage du flux $\mathcal{M}$PEG1-2 en plusieurs flux ;

* bruitage de la totalité du flux $\mathcal{M}$PEG1-2 et remise en place des *Start-Codes* et en-têtes initiaux.

[1] COdage conjoint SOurce-CAnal pour la Transmission d'Images, c'est un projet RNRT avec comme partenaires THALES, le CNES Toulouse, l'ENST Bretagne, l'ENST Paris, le L2S, France Télécom R&D et I3S.

Découpage du flux $\mathcal{M}$PEG1-2

Comme nous désirons émuler un bruit de canal (du type Rayleigh), il nous est apparu nécessaire, dans un premier temps, de savoir exactement ce que nous bruitions, c'est pour cela que nous décomposons le flux $\mathcal{M}$PEG1-2 en plusieurs flux distincts. Le premier flux sera composé des *Start-Code* et des en-têtes ; les autres flux servant à séparer les données proprement dites.

Nous allons séparer les données :

* les *VLC*[2] des *DCT* des images de type *I*,

* les *VLC* des *DCT* erreurs ou *Intra* pour les images de type *P*,

* les *VLC* des *DCT* erreurs ou *Intra* pour les images de type *B*,

* les *VLC* des vecteurs mouvements (*Forward* pour les *P* et *Forward* et/ou *Backward* pour les *B*).

Un flux sera d'autant plus altéré que les *Start-Code* et les en-têtes sont attaqués. En priorité, nous avons le *Start-Code* du *GOP* puis les informations le concernant, puis l'image *I* (c'est à partir d'elle que tout le *GOP* sera prédit), puis les *P* (les premiers *P* étant plus importants que les derniers, car ils servent pour la reconstruction du *GOP*) et enfin les *B* (certains *B* ne sont pas affichés afin de respecter la synchronisation temporelle). Nous bruitons tous les flux (*i.e.* les *VLC* des *DCT* des Intras, Prédites ou bien Bidirectionnelles et les *VLC* des vecteurs mouvements) sauf le flux *Start-Code* - en-tête.

$\mathcal{M}$PEG1-2 utilise un codage *VLC* dont le dictionnaire, pour chaque cas, est connu. Lorsque le mot qui arrive n'est pas cohérent, cela implique que celui-ci a subit une altération dû à l'ajout de bruit. Mais le fait de recevoir un mot qui existe ne veut pas dire que c'est le mot qui a été envoyé. Par exemple le mot 0000.0011.001 équivaut au mouvement -16 et le mot 0000.0011.000 équivaut, quant à lui, à +16. Les deux mots existent, sont très proches mais donnent un mouvement totalement différent.

Introduction de bruit dans les différents flux

Lorsque nous introduisons du bruit dans les *VLC* de l'Intra, cela va avoir un retentissement sur tout le *GOP* (en effet, tout le *GOP* prend appui sur cette image). Par contre, pour un *P*, le retentissement va être réduit sur les images du *GOP* qui suivent et sur les bidirectionnelles juste avant (qui prennent appui sur ce *P*). Pour finir, si c'est sur un *B*, le retentissement sera seulement sur cette image (en effet, aucune prédiction n'est faite à partir de ce type d'images). De même, lorsque nous introduisons un bruit dans les vecteurs mouvements, cela va altérer l'image et par là-même les images qui la prennent comme image pivot.

De plus, nous supposerons que nous ne bruitons pas les *Start-Code,* ni les entêtes, car avec ces informations altérées, le *GOP* peut devenir complètement indécodable. Pour ne pas bruiter cette partie, nous supposons que ce flux est envoyé dans un canal plus sûr ou bien que des éléments de redondances sont introduits afin de corriger les erreurs (turbo code ou code correcteur d'erreurs).

[2] *cf.* rappel sur la norme $\mathcal{M}$PEG1-2 vu à la page 146

Publications

Conférences internationales

ICME'03 L. Brunel, P. Mathieu, "Fast method of segmentation and indexing MPEG1-2 flow", *IEEE International Conference on Multimedia & Expo*, Baltimore, USA, juillet 2003.

VCIP'03 L. Brunel, P. Mathieu, "Fast method of segmentation and indexing MPEG1-2 flow", *SPIE Visual Communications and Image Processing*, Lugano, Suisse, juillet 2003, vol. 5150, pp. 1985-1994.

Conférences nationales

GRETSI'01 L. Brunel, P. Mathieu, « Méthode rapide de segmentation et d'indexation du flux MPEG1-2 par bloc DCT », *Groupe de Recherche et d'Étude de Traitement du Signal et des Images*, Toulouse, France, septembre 2001, volume 360.

Rapports de recherche

Rapport de D.E.A. L. Brunel, « Compression MPEG4 »,Sophia-Antipolis, France, septembre 1999.

Rapport COSOCATI M. Antonini, M. Barlaud, L. Brunel, C. Boin, T. Makram, « Transmission de vidéos à bas débits sur des canaux radio-mobiles dans le cadre de la future norme UMTS », Sophia-Antipolis, France, mars 2002.

Séminaire

GDR ISIS (Information Signal Image Vision) GT 10 L. Brunel, « Labélisation de séquences vidéos par l'utilisation du flux MPEG1 », ENST, Paris, France, mai 2001.

Bibliographie

[1] ISO/IEC JTC1 / SC29 / WG11 / M5578, *MPEG-7, visual part of experimentation model version 3.1*, décembre 1999.

[2] ISO/IEC, norme n° 11172-2, *Technologies de l'information - Codage de l'image animée et du son associé pour les supports de stockage numérique jusqu'à environ 1,5 Mbit/s : partie vidéo*, 1991.

[3] ISO/IEC, norme n° 13818-2, *Technologies de l'information - Codage générique des images animées et du son associé : partie vidéo*, 1995.

[4] N. Aboughazaleh, Y. el Gamal, "Compressed video indexing based on object motion", *SPIE Visual Communications and Image Processing*, Perth, Australie, juin 2000, pp. 986–993.

[5] O. Amadieu, E. Debreuve, M. Barlaud, G. Aubert, "Inward and outward curve evolution using level set method", *IEEE International Conference on Image Processing*, Kobe, Japon, octobre 1999, vol. 3, pp. 188–192.

[6] M. Antonini, J.-M. Moureaux, V. Lecuire, "Optimal multi-tone bit allocation for fixed rate video transmission over ADSL", *SPIE Visual Communications and Image Processing*, janvier 2002, vol. 2000.

[7] E. Ardizzone, M. la Cascia, "Video indexing using optical flow field", *IEEE International Conference on Image Processing*, Lausanne, Suisse, septembre 1996, vol. 3, pp. 831–834.

[8] G. Aubert, M. Barlaud, O. Faugeras, S. Jehan-Besson, "Image segmentation using active contours : calculus of variations or shape gradients ?", *Journal of the Society for Industrial and Applied Mathematics*, vol. 63, n° 6, pp. 2128–2154, 2003.

[9] J.L. Barron, D.J. Fleet, S.S. Beauchemin, "Performance of optical flow techniques", *International Journal of Computer Vision*, vol. 12, n° 1, pp. 43–77, février 1994.

[10] R. H. Bartels, J.C. Beatty, B.A. Barsky, *An introduction to splines for use in computer graphics and geometric modeling*, Morgan-Kaufmann Publishers, 1989.

[11] J. BENOIS, D. BARBA, "Image segmentation by region-contour cooperation for image coding", *International Conference on Pattern Recognition*, 1992, vol. C, pp. 331–334.

[12] J. Benois-Pineau, JP Braquelaire, A. Ali-Mhammad, "Interactive fine object-based segmentation of generic video scenes for object-based indexing", *European Workshop on Image Analysis for Multimedia Interactive Services WIAMIS*, Londre, Grande-Bretagne, avril 2003.

[13] J. Benois-Pineau, A. Khrennikov, "Image segmentation in compressed domain by clustering methods with euclidean and p-adic metrics", *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Annecy, France, juillet 2002, vol. III, pp. 969–976.

[14] J. Benois-Pineau, H. Nicolas, "A new method for region-based depth ordering in a video sequence : application to frame interpolation", *Journal of Visual Communications and Image Representation*, vol. 13, n° 3, pp. 363–385, septembre 2002.

[15] A.M. Bensaid, L.O. Hall, J.C. Bezdek, L.P. Clarke, "Partially supervised clustering for image segmentation", *Pattern Recognition*, vol. 29, pp. 859–871, 1996.

[16] P. Bouthemy, M. Geldon, F. Ganansia, "A unified approach to shot change detection and camera motion characterisation", *IEEE Circuits and Systems for Video Technology*, vol. 9, n° 7, pp. 1030–1044, décembre 1999.

[17] F. Brémond, M. Thonat, "Issues of representing context illustrated by video-surveillance applications", *International Journal of Human-Computer Studies. Special Issue on Context*, vol. 48, pp. 375–391, 1978.

[18] M. Cagnazzo, V. Valentin, M. Antonini, M. Barlaud, "Motion vector estimation and encoding for motion compensated dwt", *International Workshop on Very Low Bitrate Video Coding*, Madrid, Espagne, septembre 2003, pp. 233–242.

[19] P. Le Callet, D. Barba, "A robust quality metric for color image quality assessment", *IEEE International Conference on Image Processing*, Barcelone, Espagne, septembre 2003, vol. 1, pp. 437–440.

[20] M. Carnec, P. Le Callet, D. Barba, "An image quality assessment method based on perception of structural information", *IEEE International Conference on Image Processing*, Barcelone, Espagne, septembre 2003, vol. 3, pp. 185–188.

[21] V. Caselles, F. Catte, T. Coll, F. Dibos, "A geometric model for active contours", *Numerische Mathematik*, vol. 66, pp. 1–31, 1993.

[22] V. Caselles, R. Kimmel, G. Sapiro, "Geodesic active contours", *International Conference On Computer Vision*, Boston, USA, juin 1995, pp. 694–699.

[23] V. Caselles, R. Kimmel, G. Sapiro, "Geodesic active contours", *International Journal of Computer Vision*, vol. 22, n° 1, pp. 61–79, février 1997.

[24] T. Chan, L. Vese, "Active contours without edges", *IEEE Transactions on Image Processing*, vol. 10, n° 2, pp. 266–277, février 2001.

[25] J.M. Chassery, M. Melkemi, "Diagramme de voronoï appliqué à la segmentation d'images et à la détection d'évènements en imagerie multi-sources", *Traitement du signal*, vol. 8, n° 3, pp. 155–164, 1991.

[26] M. Chaumont, S. Pateux, H. Nicolas, "Segmentation of non-rigid video objects using long term temporal consistency", *IEEE International Conference on Image Processing*, Rochester, USA, septembre 2002, vol. 2, pp. 93–96.

[27] J.P. Cocquerez, S. Philipp, *Analyse d'images : filtrage et segmentation*, éditions Masson, 1995.

[28] L.D. Cohen, "On active contour models and balloons", *Computer Vision, Graphics, and Image Processing : Image Understanding*, vol. 53, n° 2, pp. 211–218, mars 1991.

[29] M. Coimbra, M. Davies, S. Velastin, "Pedestrian detection using mpeg2 motion vectors", *Workshop on Image Analysis for Multimedia Interactive Services*, Londre, GB, avril 2003, pp. 164–169.

[30] V. Coutance, *La couleur en vision par ordinateur - application à la robotique*, Thèse de doctorat, Université Paul Sabatier, Toulouse, France, janvier 1991.

[31] G. Csurka, P. Bouthemy, "Direct identification of moving objects and background from 2d motion models", *International Conference On Computer Vision*, Kerkyra, Grèce, septembre 1999, pp. 566–571.

[32] E. Debreuve, M. Barlaud, G. Aubert, J. Darcourt, "Space time segmentation using level set active contours applied to myocardial gated spect", *IEEE Transactions on Medical Imaging*, vol. 20, n° 7, pp. 643–659, juillet 2001.

[33] J.-L. Dugelay, A. Tamtaoui, B. Choquet, C. Labit, "2d and 3d motion estimation from binocular sequences by cooperation between motion and disparity in 3dtv context", *IEEE Workshop on Multidimensional Signal Processing*, Cannes, France, septembre 1993, pp. 232–233.

[34] M. Durik, J. Benois-Pineau, "Robust motion characterisation for video indexing based on mpeg2 optical flow", *International Workshop on Content-Based Multimedia Indexing*, Brescia, Italie, septembre 2001, pp. 57–64.

[35] C.L. Epstein, M. Gage, "The curve shortening flow", *Wave motion : theory, modelling, and computation (Berkeley, Calif., 1986, A. Chorin and A. Majda, Editors, Springer-Verlag)*, 1987.

[36] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design, A Practical Guide*, Academic Press Inc. 1988 existe en version française (traduction de Xavier Merlo) « Courbes et surfaces pour la Conception Géométrique Assistée par Ordinateur », édition Masson, 1988.

[37] O. Faugeras, *Three-dimensional computer vision : a geometric viewpoint*, MIT Press, Cambridge, 1999.

[38] J.D. Foley, A. Van Dam, S.K. Feiner, J.F. Hughes, *Computer graphics principles and practice*, Addison-Wesley, 1990.

[39] M. Fontaine, *Segmentation non supervisée d'images couleur par analyse de la connexité des pixels*, Thèse de doctorat, Université de Lille 1, France, décembre 2001.

[40] M. García, H. Nicolas, "Video object trajectory analysis", *IEEE International Conference on Image Processing*, Rochester, USA, septembre 2002, vol. 1, pp. 585–588.

[41] M. García, H. Nicolas, "Estimation of non-planar rotation for video coding applications", *Picture Coding Symposium*, St Malo, France, avril 2003.

[42] M. García, H. Nicolas, "Video object motion applications focusing on non-planar rotation", *IEEE International Conference on Multimedia and Expo*, Baltimore, USA, juillet 2003.

[43] M. Gastaud, M. Barlaud, "Video segmentation using active contours on a group of pictures", *IEEE International Conference on Image Processing*, Rochester, USA, septembre 2002, vol. 2, pp. 81–84.

[44] M. Gastaud, M. Barlaud, G. Aubert, "Combining shape prior and statistical features for active contour segmentation", *IEEE Transactions on Circuits and Systems for Video Technology*, mai 2004.

[45] O. Gerek, Y. Altunbasak, "Key frame selection from mpeg video data", *SPIE Visual Communications and Image Processing*, San Jose, USA, février 1997, vol. 3024, pp. 920–925.

[46] R. Hartley, A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, Cambridge, 2001.

[47] A. Herbulot, S. Jehan-Besson, M. Barlaud, G. Aubert, "Shape gradient for image segmentation using information theory", *International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Canada, mai 2004.

[48] B. Horn, *Robot vision*, MIT Press, Cambridge, 1986.

[49] S.L. Horowitz, S. Pavlidis, "Picture segmentation by a directed split and merge procedure", *IEEE International Conference on Pattern Recognition*, Copenhague, Pays-Bas, août 1974, pp. 424–433.

[50] R.W.G. Hunt, *Measuring color, applied science and industrial technology*, Ellis Horwood, 2 edition, 1991.

[51] A. E. Jacquin, "Image coding based on fractal theory of iterated contrative image transformation", *IEEE Transactions on Image Processing*, vol. 1, n° 1, janvier 1991.

[52] S. Jehan-Beson, M. Barlaud, "DREAM^{2}S : Deformable Regions Driven by an Eulerian Accurate Minimization Method for Image and Video Segmentation", *International Journal of Computer Vision*, vol. 53, n° 1, pp. 45–70, 2003.

[53] S. Jehan-Besson, *Modèles de contours actifs basés régions pour la segmentation d'images et de vidéo*, Thèse de doctorat, Université Nice - Sophia Antipolis, France, 6 janvier 2003.

[54] S. Jehan-Besson, M. Barlaud, G. Aubert, "Video object segmentation using eulerian region-based active contours", *International Conference On Computer Vision*, Vancouver, Canada, 2001.

[55] S. Jehan-Besson, M. Barlaud, G. Aubert, "A 3-step algorithm using region-based active contours for video objects detection", *EURASIP Journal on Applied Signal Processing*, vol. 2002, n° 6, pp. 572–581, juin 2002.

[56] S. Jehan-Besson, M. Gastaud, F. Precioso, M. Barlaud, E. Debreuve, G. Aubert, "From snakes to region based active contours", *Applied Optics*, vol. 43, n° 2, pp. 247–256, janvier 2004.

[57] K. Jinzenji, S. Ishibashi, H. Kotera, "Algorithm for automatically producing layered sprites by detecting camera movement", *IEEE International Conference on Image Processing*, Washington DC, USA, octobre 1997, pp. 767–770.

[58] J. Jung, M. Antonini, M. Barlaud, "Décodage optimal orienté objet pour la suppression des effets de bloc dans les séquences dv et mpeg2", *GRETSI Conference on Signal and Image Process*, Vannes, France, septembre 1999.

[59] J. Jung, M. Antonini, M. Barlaud, "Optimal decoder for block transform based video coders", *IEEE Transactions on Multimedia*, vol. 5, n° 2, pp. 145–160, juin 2003.

[60] M. Kass, A. Witkin, D. Terzopoulos, "Snakes : Active contour models", *International Journal of Computer Vision*, vol. 1, n° 4, pp. 321–331, 1988.

[61] G.J. Klinker, S.A. Shafer, T. Kanade, "A physical approach to color image understanding", *International Journal of Computer Vision*, vol. 4, n° 1, pp. 7–38, 1990.

[62] M. Kunt, G. Granlund, M. Kocher, *Traitement numérique d'images*, Presses Polytechniques et universitaires romandes, 1993.

[63] B. Lo, S.A. Velastin, M.A. Vicencio-Silva, J. Sun, "An intelligent distributed surveillance system for public transport", *Distributed surveillance systems (IDSS)*, février 2003.

[64] A. Mahboubi, J. Benois-Pineau, D. Barba, "Tracking of objects in video scenes with time varying content", *EURASIP Journal of Applied Signal Processing (JASP)*, vol. 2002, n° 6, pp. 582–594, juin 2002.

[65] R. Malladi, J.A. Sethian, B.C. Vemuri, "Shape modeling with front propagation : a level set approach", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, n° 2, pp. 158–175, février 1995.

[66] R. Malladi, J.A. Sethian, B.C. Vemuri, "A fast level set based algorithm for topology-independent shape modeling", *Journal of Mathematical Imaging and Vision, Special issue on Topology and Geometry*, vol. 6, n° 2/3, pp. 269–290, 1996.

[67] A.-R. Mansouri, B. Sirivong, J. Konrad, "Multiple motion segmentation with level sets", *SPIE Image and Video Communications and Process.*, vol. 3974, pp. 584–595, janvier 2000.

[68] M. de Marsicoi, L. Cinque, S. Levialdi, "Indexing pictorial documents by their content : a survey of current techniques", *Image and Vision Computing*, vol. 15, n° 2, pp. 119–141, février 1997.

[69] F.X. Martinez, J. Benois-Pineau, D. Barba, "Extraction of the relative depth information of objects in vidéo sequences", *IEEE International Conference on Image Processing*, Chicago, USA, octobre 1998, pp. 948–952.

[70] B.A. Maxwell, S.A. Shafer, "Physics-based segmentation : looking beyond color", *Image Understanding Workshop*, Palm Springs, USA, février 1996, pp. 867–878.

[71] P. Migliorati, S. Tubaro, "Multistage motion estimation for image interpolation", *EURASIP journal on Applied Signal Processing : Image Communication*, vol. 7, pp. 187–199, 1995.

[72] M. Miyahara, Y. Yoshida, "Mathematical transform of (r,g,b) color data to munsell (h,s,v) color data", *SPIE Visual Communications and Image Processing*, San-Jose, USA, 1988, vol. 1001, pp. 650–657.

[73] H. Miyamori, "Automatic annotation of tennis action for content-based retrieval by integrated audio and visual information", *International Conference on Image and Video Retrieval*, Urbana-Champaign, USA, juillet 2003, pp. 331–341.

[74] F. Morier, J. Benois-Pineau, D. Barba, "Robust segmentation of moving image sequences", *IEEE International Conference on Image Processing*, Washington DC, USA, octobre 1997, pp. 555–558.

[75] F. Morier, H. Nicolas, J. Benois-Pineau, D. Barba, H. Sanson, "Relative depth estimation of video objects for image interpolation", *IEEE International Conference on Image Processing*, Chicago, USA, octobre 1998, pp. 953–957.

[76] J. Motsch, H. Nicolas, "Classification de mouvement des objet", *GRETSI Conference on Signal and Image Process*, Vannes, France, septembre 1999, pp. 183–186.

[77] C. Nastar, M. Mitschke, "Real-time face recognition using feature combination", *IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japon, avril 1998.

[78] R. Nevatia, "A colour edge detector and its use in scene segmentation", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, n° 11, pp. 820–826, novembre 1977.

[79] H. Nicolas, "New methods for dynamic mosaicking", *IEEE Transactions on Image Processing*, vol. 10, n° 1, pp. 1239–1251, août 2001.

[80] H. Nicolas, F. Denoual, "Interactive modifications of video object trajectories in natural video sequences for post-production applications", *IEEE International Conference on Image Processing*, Rochester, USA, septembre 2002, vol. 2, pp. 325–328.

[81] H. Nicolas, C. Labit, "Global motion identification for image sequence analysis and coding", *IEEE Conference on Acoustics, Speech and Signal Processing*, Toronto, Canada, mai 1991, vol. 4, pp. 2825–2828.

[82] H. Nicolas, C. Labit, "Region-based motion estimation using deterministic relaxation schemes for image sequence coding", *IEEE Conference on Acoustics, Speech and Signal Processing*, San-Francisco, USA, mars 1992, vol. 3, pp. 265–268.

[83] J.M. Odobez, *Estimation, detection et segmentation du mouvement : une approche robuste et markovienne*, Thèse de doctorat, Thèse de l'Université de Rennes I, France, décembre 1994.

[84] N. Ohta, "Correspondance between cielab and cieluv color differences", *Color Research and Application*, vol. 2, n° 4, pp. 178–182, 1977.

[85] Y.I. Ohta, T. Kanade, T. Sakai, "Color information for region segmentation", *Computers Graphics and Image Processing*, vol. 13, n° 2, pp. 222–241, 1980.

[86] A. Pentland, R.W. Picart, S. Sclaroff, "Photobook : tools for content-based manipulation of image databases", *International Journal of Computer Vision*, vol. 18, n° 3, pp. 233–254, 1996.

[87] F. Perronnin, J.-L. Dugelay, K. Rose, "Deformable face mapping for person identification", *IEEE International Conference on Image Processing*, Barcelone, Espagne, septembre 2003, vol. 1, pp. 661–664.

[88] F. Precioso, M. Barlaud, "B-Spline active contour for fast video segmentation", *European Workshop on Image Analysis for Multimedia Interactive Services WIAMIS*, Tempere, Finlande, mai 2001.

[89] F. Precioso, M. Barlaud, "B-Spline active contour with handling of topological changes for fast video segmentation", *EURASIP journal on Applied Signal Processing Special issue on Image Analysis for Multimedia Interactive Services*, vol. 2002, n° 6, pp. 555–560, juin 2002.

[90] F. Precioso, M. Barlaud, "Regular spatial B-Spline active contour for fast video segmentation", *IEEE International Conference on Image Processing*, Rochester, USA, septembre 2002, vol. 2, pp. 761–764.

[91] F. Precioso, M. Barlaud, T. Blu, M. Unser, "Smoothing B-Spline active contour for fast and robust image and video segmentation", *IEEE International Conference on Image Processing*, Barcelona, Spain, septembre 2003, vol. 1, pp. 137–140.

[92] F. Precioso, M. Barlaud, T. Blu, M. Unser, "Smoothing B-Spline active contour for fast and robust image and video segmentation", *IEEE Transactions on Image Processing*, 2004.

[93] A.M. Rohaly, P. Corriveau, J. Libert, A. Webster, V. Baroncini, J. Beerends, J.L. Blin, L. Contin, T. Hamada, D. Harrison, A. Hekstra, J. Lubin, Y. Nishida, R. Nishihara, J. Pearson, A. Franca Pessoa, N. Pickford, A. Schertz, M. Visc, A. Watson, S. Winkler, "Video quality experts group : Current

results and future directions", *SPIE Visual Communications and Image Processing*, Perth, Australie, juin 2000, vol. 4067, pp. 742–753.

[94] C. Rosenberger, "Fusion génétique de résultats de segmentation", *GRETSI Conference on Signal and Image Processing*, Toulouse, France, septembre 2001.

[95] J.M. Rubin, W.A. Richards, "Color vision : representing material changes", *AI Memo 764, MIT Artificial Intelligence Laboratory*, mai 1984.

[96] G. Sapiro, *Geometric partial differential equations and image analysis*, Cambridge university press, 2001.

[97] I.J. Schoenberg, "Contribution to the problem of approximation of equidistant data by analytic functions", *Quart. Appl. Math*, vol. 4, pp. 45–99 112–141, 1946.

[98] H. Senane, A. Saadane, D. Barba, "Image coding in the context of a psychovisual image representation with vector quantization", *IEEE International Conference on Image Processing*, Washington DC, USA, octobre 1995, vol. 1, pp. 97–100.

[99] Y. Seo, S. Choi, Y. Kim, K.S. Hong, "Where are the ball and player ? soccer game analysis with color-based tracking and image mosaik", *International Conference on Image Analysis and Processing*, Florence, Italie, 1997, vol. 2, pp. 196–203.

[100] K. Siddiqi, Y.B. Lauzière, A. Tannenbaum, S.W. Zucker, "Area and length minimizing flows for shape segmentation", *IEEE Transactions on Image Processing*, vol. 7, n° 3, pp. 433–443, mars 1998.

[101] J.R. Smith, *Integrated Spatial and Feature Image Systems : retrieval, Analysis and Compression*, Thèse de doctorat, Colombia University, USA, 1997.

[102] M. V. Srinivasan, S. Venkatesh, R. Hosie, "Qualitative estimation of camera motion parameters from video sequences", *Pattern Recognition*, vol. 30, pp. 593–606, avril 1997.

[103] J. Stauder, H. Nicolas, "Motion-based video indexing evaluating object shading", *IEEE International Conference on Image Processing*, Kobe, Japon, octobre 1999, vol. 3, pp. 265–268.

[104] C. Stiller, J. Konrad, "Estimating motion in image sequences", *IEEE Signal Processing Magazine*, vol. 16, pp. 70–91, juillet 1999.

[105] G. Sudhir, A.K. Jain J.C.M. Lee, "Automatic classification of tennis video for high-level content-based retrieval", *International Workshop on Content-Based Access of Image and Video Databases*, Bombay, Inde, janvier 1998, p. 81.

[106] M.J. Swain, C. Frankel, V. Athitsos, "Webseer : an image search engine for the world wide web", *IEEE Computer Vision and Pattern Recognition Conference , year = 1997, address = San Juan, Puerto Rico, month = jun.*

[107] Y.P. Tan, S.R. Kulkarni, P.J. Ramadge, "A new method for camera motion parameter estimation", *IEEE International Conference on Image Processing*, Washington DC, USA, octobre 1995, vol. 1, pp. 406–409.

[108] Y.P. Tan, D.D. Saur, S.R. Kulkarni, P.J. Ramadge, "Rapid estimation of camera motion from compressed video with application to video annotation", *IEEE Transactions Circuits and systems for video technology*, vol. 10, n° 1, pp. 133–146, février 2000.

[109] A. Trémeau, N. Borel, "Region adjacency graph applied to color image segmentation", *IEEE Transactions on Image Processing*, vol. 9, n° 4, pp. 735–744, avril 2000.

[110] E.C.K. Tsao, J.C. Bezdek, N.R. Pal, "Fuzzy kohonen clustering networks", *Pattern Rocognition*, vol. 27, n° 5, pp. 757–764, mai 1994.

[111] T. Uchiyama, M.A. Arbib, "Color image segmentation using competitive learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, n° 12, pp. 1197–1206, décembre 1994.

[112] M. Vissac, J.-L. Dugelay, K. Rose, "A fractals-inspired approach to content-based image indexing", *IEEE International Conference on Image Processing*, Kobe, Japon, octobre 1999, vol. 2, pp. 575–579.

[113] M. Vissac, J.-L. Dugelay, K. Rose, "A novel indexing approach for multimedia image databases", *IEEE Signal Processing Society 1999 Workshop on Multimedia Signal Processing*, Copenhague, Danemark, septembre 1999, pp. 97–102.

[114] J. Wang, D. Barba, "Object-oriented motion estimation for image sequence coding", *IEEE Workshop on Multimensional Signal Processing*, Cannes, France, septembre 1993, pp. 234–234.

[115] R. Wang, T. Huang, "Fast camera motion analysis in mpeg domain", *IEEE International Conference on Image Processing*, Kobe, Japon, octobre 1999, vol. 3, pp. 691–694.

[116] S.S. Wong, W.K. Leow, "Color segmentation and figure-ground segregation of natural images", *IEEE International Conference on Image Processing*, Vancouver, Canada, septembre 2000, p. TA04.08.

[117] G. Wyszecki, W. S. Stiles, *Color Science : concepts and methods, quantitative data and formulae*, Wiley Classics Library, second edition, 1982.

[118] J.C. Yen, F.J. Chang, S. Chang, "A new criterion for automatic multilevel thresholding", *IEEE Transactions on Image Processing*, vol. 4, n° 3, pp. 370–378, mars 1995.

[119] P.Y Yin, L.H. Chen, "Random sampling thresholding : a new approach to multilevel thresholding", *Signal Processing*, vol. 34, n° 3, pp. 311–322, 1993.

[120] S. Zhu, A. Yuille, "Region competition : unifying snakes, region growing, and bayes/mdl for multiband image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 884–900, septembre 1996.

[121] S.W. Zucher, "Region growing : childhood and adolescence", *Computer Graphics and Image Processing*, vol. 5, pp. 382–399, 1976.

Résumé : « Indexation vidéo par l'analyse de codage »

Ce travail de thèse porte sur l'indexation, normalisée par $\mathcal{M}$PEG7, de séquences vidéos. À partir d'un flux $\mathcal{M}$PEG1-2, ou de tout autre codec basé sur la prédiction de mouvement et la *DCT*, sans totalement le décompresser, nous exploitons l'analyse effectuée lors du codage. Ainsi de façon non-supervisée et en quasi temps réel, nous proposons une méthode d'estimation du mouvement de la caméra ainsi que d'extraction des objets en déplacement.

Pour l'estimation du mouvement de la caméra, nous utilisons les vecteurs de prédiction temporelle présents dans le flux. L'étude des images d'erreur nous permet d'en évaluer la pertinence.

Pour la détection des objets en mouvement, nous segmentons tout d'abord la séquence en zones de couleur uniforme directement sur les coefficients *DCT*. Nous établissons une distance colorimétrique, non seulement entre deux pixels voisins d'une image, mais aussi entre deux images successives, ce qui définit une zone en trois dimensions. Afin de segmenter plus précisement et de régulariser les contours sur chaque image, nous utilisons les *B-Splines*. Chaque objet candidat est déformé par la présence de tous ses voisins à partir d'un potentiel de couleur, ce qui, itérativement, permet d'éliminer les zones trop réduites.

En combinant le mouvement de la caméra, les vecteurs de prédiction et les zones de couleur 2D+t, nous réalisons une fusion adaptative de façon à obtenir une bonne représentation des objets.

Mots clefs : Traitement vidéo - estimation mouvement caméra - détermination zones de couleur - régularisation par *B-Splines* - segmentation objets mobiles - distance colorimétrique - analyse du flux $\mathcal{M}$PEG1-2 - $\mathcal{M}$PEG7

Abstract : "Video indexation by encoding analysis"

This thesis work concerns indexation, normalized by $\mathcal{M}$PEG7, of video sequences. From a $\mathcal{M}$PEG1-2 stream, or from any other codec based on movement prediction and DCT, without decompressing it completely, we exploit the analysis carried out during the encoding process. This way, unsupervised and in quasi real-time, we provide a method to estimate the camera movement as well as moving objects extraction.

As far as camera movement estimation is concerned, we use motion vectors included in the stream. Studying the error images allows us to assess its relevance.

In order to detect moving objects, we first segment the sequence into uniform color zones directly on the DCT coefficients. We establish a colorimetric distance, not only between two neighbouring pixels inthe same image, but also in two successive images, which allows to define a three-dimensional zone. In order to provide a more accurate segmentation, and to regularize the contours on every image, we use B-splines. Every candidate object is distorted by the presence of all its neighbors, based on a color potential. This allows iteratively to eliminate zones which were excessively reduced.

By combining camera movements, prediction vectors and 2D+t color zones, we create an adaptative fusion in order to obtain a good representation of objects, and thus of their monitoring.

Keywords : Video processing - camera movement estimation - color zones determination - B-Splines regularization - moving objects segmentation - color distance - $\mathcal{M}$PEG1-2 stream analysis - $\mathcal{M}$PEG7